Trabajo Fin de Máster

Optimización del transporte de muestras biomédicas

Ángel Martínez González

Máster en Técnicas Estadísticas

Curso 2021-2022

Propuesta de Trabajo Fin de Máster

Título en galego: Optimización do transporte de mostras biomédicas
Título en español: Optimización del transporte de muestras biomédicas
English title: Optimization in Biomedical Sample Transportation
Modalidad: Modalidad A
Autor: Ángel Martínez González, Universidad de Santiago de Compostela
Directora: Balbina Virginia Casas Méndez, Universidad de Santiago de Compostela
Breve resumen del trabajo: Revisión y aplicación del problema de optimización del transporte de muestras biomédicas. Plantearemos una descripción del estado del arte correspondiente a este problema para realizar una selección de modelos y abordar un caso práctico tomado de la vida real. Interesa evaluar el rendimiento de estos modelos y concluir el tamaño de problemas que pueden ser resueltos de manera exacta o aproximada y analizar los tiempos de computación según el método usado. Además, consideraremos el diseño e implementación de un algoritmo que permita la resolución de problemas de mayor tamaño.
Recomendaciones: Conocimiento del uso de los software AMPL y R.

Doña Balbina Virginia Casas Méndez, profesora titular de la Universidad de Santiago de Compostela, informa que el Trabajo Fin de Máster titulado

Optimización del transporte de muestras biomédicas

fue realizado bajo su dirección por don Ángel Martínez González para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, da su conformidad para su presentación y defensa ante un tribunal.

En Santiago de Compostela, a 30 de agosto de 2022.

CASAS MENDEZ BALBINA VIRGINIA - 09367287 P

Firmado digitalmente por CASAS MENDEZ BALBINA VIRGINIA - 09367287P
Fecha: 2022.08.30 23:09:04 +02'00'

La directora:
Doña Balbina Virginia Casas Méndez

El autor:
Don Ángel Martínez González

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el autor declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,... sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Agradecimientos

Tengo que agradecer a mi familia por todo el apoyo dado de distintas maneras en estos años tanto del Grado en Matemáticas como en el presente Máster en Técnicas Estadísticas, a los compañeros que me han brindado apoyo y ayudado a hacer estos años más llevaderos en los momentos difíciles, a los profesores que me han ayudado a llegar hasta aquí y a mi tutora de TFM Balbina Casas Méndez, con la cual he podido contar para hacer de este trabajo lo que es hoy.

Índice general

Resumen

Resumen en español

Revisión y aplicación del problema de optimización del transporte de muestras biomédicas, centrando la aplicación del mismo en el área sanitaria de Santiago de Compostela y Barbanza (ASSC). Después de la presentación de las características de esta zona y el funcionamiento del transporte y clasificación de las muestras, se analiza brevemente la literatura previa que nos lleva a la propuesta de esta revisión, así como la literatura previa sobre estudios análogos a nuestro problema. Todo esto será realizado para dar una presentación del modelo con el que trabajaremos. Posteriormente a la formulación de este problema, se abordará la resolución del mismo tanto de manera exacta como heurística con el objetivo de ayudar a optimizar las rutas actuales. El método heurístico propuesto estará basado en ideas muy sencillas, buscando observar su efectividad. Finalmente, se presentan resultados obtenidos de cada forma y se compararán con las rutas que existen actualmente.

English abstract

A review and implementation of the biomedical sample transportation problem, focussed on its implementation in the Santiago de Compostela and Barbanza Health Area (SCHA). After introducing the characteristic of this place and how the transport and classification work, previous literature about the fact which inspires this revision will be analysed briefly. Also the state-of-art will be included. All this previous studies will help us in order to introduce the formal problem description we are going to work with. After that, the solution of the problem will be studied in two ways to optimize the current routes: With an exact solution and with an heuristic solution. This heuristic solution will be based on very simple ideas, in order to observe their efficacy. In the end, some results obtained in this differents ways will be introduced and compared with the real routes existing nowdays.

Prefacio

Los problemas de optimización del transporte de muestras biomédicas son una aplicación muy importante en el campo de la investigación operativa, pues tratan, entre otros casos, de buscar mejoras en las rutas de vehículos que se encargan de llevar las muestras biomédicas a partir de distintos centros de salud para su análisis en el laboratorio. La planificación de estas rutas es importante, pues un mal diseño, además de otros factores externos, pueden deteriorar las muestras y dar lugar a análisis erróneos. Si bien hay muchas formas de estudiar y analizar este tipo de problemas, en este trabajo haremos un enfoque completamente práctico del mismo, buscando ver las limitaciones que tiene intentar resolver un problema de este estilo de forma exacta y dando también las bases de un método heurístico. Para ello, centraremos la resolución de este problema en el área sanitaria de Santiago de Compostela y Barbanza.

En el primer capítulo presentaremos las características geográficas, el funcionamiento del transporte de las muestras en detalle y la literatura previa que nos ha traído hasta pensar en este análisis. En el capítulo dos se presentarán problemas propuestos anteriormente y relacionados con el nuestro, cuya formulación será presentada en el capítulo tres. El cuarto y quinto capítulo buscan dar soluciones a este problema de forma exacta y mediante un método heurístico, respectivamente. Finalmente presentaremos los resultados obtenidos por cada método, los compararemos con las rutas que se realizan actualmente y terminaremos con una discusión sobre los resultados obtenidos y las mejoras que se pueden implementar con respecto al método heurístico aquí presentado.

Capítulo 1

Transporte de muestras en el área sanitaria de Santiago de Compostela

En este primer capitulo comenzaremos a explicar la situación del área sanitaria de Santiago de Compostela (véase Figura 1), así como el caso del transporte de muestras biomédicas que nos ha traído hasta el análisis del problema de optimización con el que vamos a trabajar. También analizaremos los objetivos a cumplir en dicho problema.

1.1. Datos sobre el área sanitaria de Santiago de Compostela

- **Situación:** El área sanitaria de Santiago de Compostela (ASSC) se encuentra en Galicia, al noroeste de España.
- **Extensión:** Ocupa un área de 4905 km^2 aproximadamente, cubriendo un total de 46 municipios.
- **Población:** Un total de $442,897$ usuarios[1] tienen acceso a los servicios ofrecidos a fecha de la entrega del presente trabajo.
- **Hospital de referencia:** El Hospital Clínico Universitario de Santiago (CHUS) es el centro de referencia con respecto al resto de centros sanitarios del ASSC. Se encuentra situado en Santiago de Compostela.
- **Centros sanitarios:** Un total de 69 centros sanitarios asociados al Servicio Gallego de Salud (SERGAS), además del CHUS, el cual los coordina, forman parte de esta área.
- **Distancias:** Los centros sanitarios del ASSC se encuentran, a lo sumo, a unos 70 km del CHUS.

[1] Acceso a datos Santiago
Acceso a datos Barbanza

- **Muestras biomédicas:** De lunes a viernes se realizan recogidas de muestras de sangre en los distintos hospitales y centros de salud del ASSC para su posterior análisis en el CHUS.

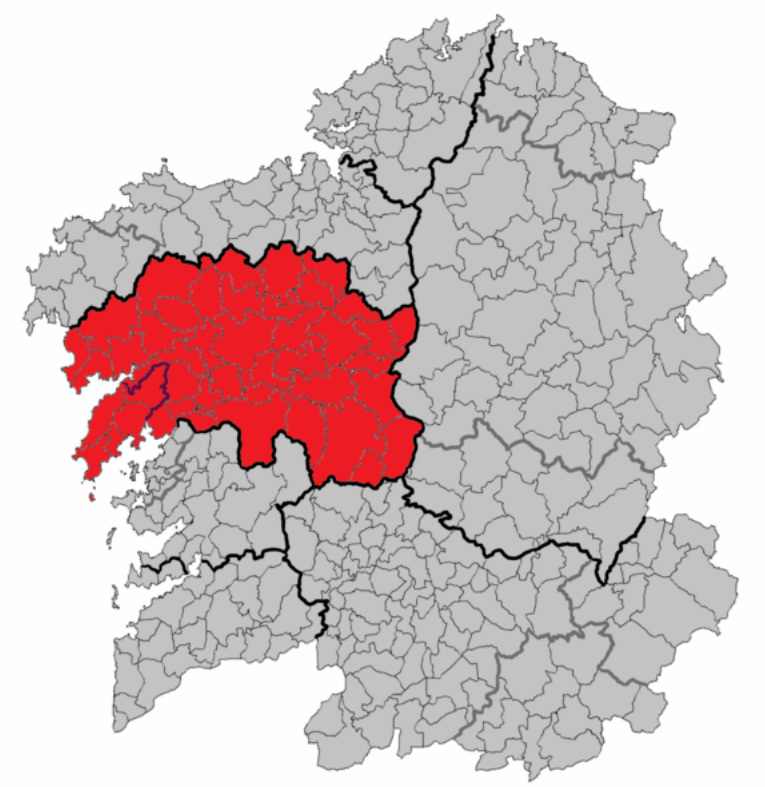


Figura 1: Mapa de las áreas sanitarias de Galicia. En rojo, destacada, el área del ASSC, objeto de nuestro estudio.

1.2. El proceso de análisis

1.- **El transporte:** Un conjunto de ambulancias de una empresa subcontratada se encargan de transportar las muestras desde los distintos centros del ASSC hasta el CHUS.

2-. **La recepción:** Al llegar una ambulancia al CHUS, las muestras transportadas (que pueden ser de varios tipos) son entregadas en recepción.

3.- **La separación:** Dos personas se encargan de quedarse con las muestras de sangre (el mayor porcentaje de muestras), y separarlas de las muestras que se analizan en otros departamentos (como las de orina). Estas personas colocan las muestras en bandejas para llevarlas al mecanismo denominado como MUT, que se encarga de separarlas y clasificarlas.

4.- **La clasificación:** El MUT se encarga de clasificar las muestras, mediante sus colores, en distintos compartimentos, dejando registro de la hora en la que se escanea la muestra. A partir de aquí, las muestras realizan el camino correspondiente a la prueba que se les debe realizar.

1.3. Diferencias geográficas

En el artículo de Espasandín-Domínguez et al. (2018) se realiza un análisis de la concentración de potasio sérico registrada y su variabilidad, para analizar si hay una posible diferencia de concentración dependiendo de la zona de la que provengan las muestras de sangre transportadas en el ASSC. Esta variabilidad de concentraciones puede atribuirse a factores preanalíticos que pueden influir en el resultado del análisis. Para el estudio, se ha utilizado un modelo de regresión distribucional aditivo estructurado.

En este trabajo se detectan muestras con altas concentraciones de potasio sérico, pero todas resultan ser de zonas, en general, alejadas del CHUS, lo que da lugar a un problema de *pseudohyperkalemia*, es decir, que sean resultado de una gestión de la muestra no del todo buena. Esto no es necesariamente una crítica a las personas encargadas de obtener y transportar la misma, pues la lista de factores que pueden modificar los resultados es amplia.

En base a este estudio se han propuesto varias soluciones para evitar estos problemas como puede ser una mejor práctica de laboratorio. Otra solución, la cual es la que analizaremos aquí, es una mejora en el procedimiento del transporte de las muestras. Más concretamente, buscaremos mejorar las rutas de transporte, de manera que las muestras puedan llegar antes al CHUS y se protejan más, evitando, de esta forma, el deterioro de las mismas.

1.4. Los problemas del transporte

El transporte en el ASSC comienza a las 8.00 horas y cuenta en la actualidad con 7 rutas de lunes a viernes excepto los miércoles, día en el que tenemos una ruta extra como apoyo, dando un total de 8 rutas en este día de la semana. Además, la carga en los distintos vehículos no se traducirá en una restricción de límite de capacidad relevante.

Tenemos varios aspectos que podríamos cambiar para mejorar el transporte de las muestras. En primer lugar, vendría bien una replanificación de las rutas. Las rutas que se siguen ahora mismo son rutas propuestas en cierto momento, pero que han podido quedarse obsoletas con la aparición de nuevos centros médicos. Por lo tanto, sería conveniente una revisión de estas rutas para la correcta inclusión de los nuevos centros en las mismas perjudicando en la menor manera posible la llegada al CHUS.

Por otro lado, están las llegadas al CHUS. El problema se produce cuando varias ambulancias llegan a la vez, provocando colas y llegando a colapsar el sistema. Por lo tanto, sería recomendable que los vehículos llegasen al CHUS con espacio de tiempo entre ellos para que no se deterioren más de lo que sea inevitable en el transporte. Se tiene conocimiento de las muestras transportadas en cada ruta y el número de éstas por unidad de tiempo que cada persona de la recepción del CHUS puede gestionar desde su llegada hasta que se dejan en el MUT. Por consiguiente, esto nos permite determinar una cantidad de tiempo mínima adecuada entre la llegada de dos ambulancias consecutivas.

Por último, y el problema principal que nos lleva a este planteamiento, es la duración de las rutas. Como hemos comentado, las muestras de las zonas más alejadas del CHUS producían

muestras con *pseudohyperkalemia*, señalando erróneamente un problema en el organismo en niveles de potasio. Por lo tanto, sería recomendable procurar buscar una forma de reducir los tiempos de estas rutas para evitar que aparezcan estos falsos datos.

1.5. El problema a solucionar. Análisis similares

En conclusión, lo que buscaremos solucionar es un problema de transporte de muestras donde la meta es minimizar el tiempo total de transporte poniendo límites a la duración de cada ruta e imponiendo una llegada ordenada de los vehículos al CHUS, es decir, considerando ventanas de tiempo. La propuesta y el análisis inicial para el problema en el ASSC vienen dados en Casas-Méndez y Davia-Pena (2021a).

Como profundizaremos en el siguiente capítulo, se pueden encontrar artículos de problemas parecidos en la literatura, dándonos una idea de lo que deberemos realizar para afrontar nuestro cometido. Por ejemplo, Yi (2003) o Yücel et al. (2013) estudian de maneras distintas problemas de recogida y envío .

Con respecto a un problema muy cercano al nuestro, tenemos el denominado problema de transporte de muestras biomédicas, sobre el cual Anaya-Arenas et al. (2019) realizan un estudio motivado por la situación de la sanidad de Quebec, Canadá.

Capítulo 2

Estado del arte

En este capítulo nos centraremos en revisar distintos modelos de investigación operativa que tienen relación con el objetivo final de este trabajo para, finalmente, tratar de modelizar el problema de interés y trabajar con este caso.

2.1. El problema de rutas de vehículos (VRP)

El problema de rutas de vehículos, o por su título en inglés *Vehicle Routing Problem* (VRP), trata de encontrar un conjunto de rutas desde un lugar de origen al que regresan habiendo pasado, entre todos, por las distintas localizaciones de atención cumpliendo con las demandas en cada una de estas localizaciones. La primera aparición en la literatura de este problema se asocia a George Dantzig (1914-2005) y John Ramser, los cuales, en 1959, lo presentan junto a un algoritmo de resolución aplicado a un problema de distribución entre gasolineras, ilustrando una solución con un ejemplo ficticio para enseñar su funcionamiento. Posteriormente, otros estudios han ido mejorando esta primera resolución, como Clarke y Wright en 1964 con un método heurístico.

Pasaremos a la explicación del problema, empezando con los conjuntos, parámetros, variables necesarias, y viendo, posteriormente, las restricciones y objetivos que determinan este modelo.

2.1.1. Conjuntos, parámetros y variables

- $G = (V, A)$ es un grafo donde:
  - $V = \{0, 1, ..., n, n+1\}$ es el conjunto de vértices, de los cuales $N = \{1, .., n\}$ representan las localizaciones de los clientes, y siendo los valores 0 y $n+1$ representantes del lugar de salida y llegada de todos los vehículos (el almacén).
  - $A = \{(i, j)/i \neq j, i, j \in V\}$ es el conjunto de arcos que unen los vértices. Obsérvese que, por la definición del conjunto, los nodos i y j son distintos, es decir, no se consideran rutas directas de un nodo a sí mismo. Asociado a cada uno de estos arcos, tenemos el parámetro c_{ij}, el cual se interpreta como el coste (en dinero, tiempo, etc.) de ir desde

el nodo $i \in V$ al nodo $j \in V$. El conjunto de estos costes se establece como una matriz, a la que llamamos C.

- $M = \{1, ..., m\}$ es el conjunto de vehículos disponibles para la realización de las rutas. Se establece que los costes asociados a los vehículos van a ser iguales sin importar el vehículo usado de este conjunto, es decir, no tendremos costes adicionales por el uso de uno u otro vehículo
- Q nos indica la capacidad máxima del vehículo. En este problema, todos los vehículos tendrán la misma capacidad máxima.
- q_i indica la cantidad a entregar al cliente $i \in V$.
- x_{ijk} es una variable binaria que vale 1 si el vehículo $k \in M$ va de $i \in V$ a $j \in V$ y 0 en otro caso.
- $\delta^+(i) = \{j \in N/(i,j) \in A\}$ es el conjunto de arcos de salida del nodo $i \in N$ y $\delta^-(j) = \{i \in N/(i,j) \in A\}$ el conjunto de arcos de entrada del nodo $j \in N$.

2.1.2. Objetivo y restricciones

El objetivo de este problema será el siguiente:

$$min \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} \sum_{k=1}^{m} c_{ij} x_{ijk}$$

Para buscar la solución, el problema requiere de las siguientes restricciones:

1. Cada vehículo debe ser visitado una única vez y por un único vehículo:

$$\sum_{k \in M} \sum_{j \in \delta^+(i)} x_{ijk} = 1 \quad i \in N$$

2. El vehículo que pasa por una localización también debe abandonarla:

$$\sum_{i \in \delta^-(j)} x_{ijk} - \sum_{i \in \delta^+(j)} x_{jik} = 0 \quad j \in N, \quad k \in M$$

3. Comienzo de ruta desde el almacén:

$$\sum_{j \in \delta^+(0)} x_{0jk} = 1 \quad k \in M$$

4. Final de ruta también en el almacén:

$$\sum_{i \in \delta^-(n+1)} x_{i(n+1)k} = 1 \quad k \in M$$

5. Los vehículos sólo son utilizados una vez:

$$\sum_{j=0}^{n} x_{0jk} = 1 \quad k \in M$$

6. La capacidad de los vehículos es limitada:

$$\sum_{i \in N} q_i \sum_{j \in \delta^+(i)} x_{ijk} \leq Q \quad k \in M$$

Para acabar con esta formulación, añadir que, a priori, podrían formarse ciclos, es decir, uniones entre nodos que no estén conectados a nuestros nodos almacén. Para ello, podemos contar con distintas restricciones que eviten esto, como la reformulación de *Miller-Tucker-Zemlin* (MTZ), con variables adicionales (nodos potenciales) que eliminan la posibilidad de subciclos, o la reformulación de flujo de red (NFB), en la cual tenemos variables adicionales por cada arco que fuerzan el flujo de la manera deseada.

2.2. Problema de recogida y envío

Se trata de una variante del problema del viajante, en la cual un conjunto de bienes deben ser transportados desde su localización inicial hasta los lugares de entrega necesarios. Existen varios tipos, incluyendo problemas donde la carga de un nodo debe ir necesariamente a otro o problemas donde el objetivo es satisfacer las demandas de los puntos de entrega, sin necesidad de provenir de un origen concreto.

En este problema distinguiremos los nodos de carga de los nodos de entrega, teniendo que establecer los nodos relacionados con los mismos vehículos. De esta forma, los elementos del problema son como siguen. Después procederemos a dar la formulación del problema.

2.2.1. Conjuntos, parámetros y variables

- $G = (V, A)$ es un grafo donde:
  - $V = \{0, 1, ..., n, n+1, ..., 2n, 2n+1\}$ es el conjunto de vértices, de los cuales $P = \{1, .., n\}$ representan las localizaciones de carga, $D = \{n+1, .., 2n\}$ son las localizaciones de entrega, asociando $n+i$ a la carga de $i \in P$, y siendo los valores 0 y $2n+1$[1] representantes del lugar de salida y llegada de todos los vehículos (el almacén). $\forall i \in N$ tenemos también:
    - Los tiempos de duración del servicio correspondiente en el nodo i, d_i.
    - Al utilizar ventanas de tiempo, tenemos los tiempos mínimos de llegada y máximos de salida, e_i y l_i, respectivamente.
    - Las cantidades que se cargan o se demandan en cada nodo, con el parámetro q_i. En particular:
      - $q_i > 0 \Rightarrow i \in P$ nodo de recogida.

[1] En una definición del problema más general, no es necesario que las cantidades de nodos de carga y entrega sean iguales

- $\diamond$ $q_i < 0 \Rightarrow i \in D$ nodo de entrega.
- $\diamond$ $q_i = 0 \Rightarrow i \in \{0, 2n+1\}$ nodos almacén.

- $A = \{(i,j)/i \neq j, i, j \in V\}$ es el conjunto de arcos que unen los vértices.

- $M = \{1, ..., m\}$ es el conjunto de vehículos disponibles para la realización de las rutas. Asociado a cada vehículo, tenemos:
  - Un tiempo límite T_k, que viene dado por el tiempo de trabajo posible para cada vehículo.
  - Una capacidad máxima para cada vehículo, representada por C_k.
  - Asociado a cada uno de los arcos antes definido, tenemos su tiempo de trayecto t_{ijk} y coste c_{ijk} que representan los valores de tiempo y dinero, respectivamente, en ir desde el nodo $i \in V$ al nodo $j \in V$ con el vehículo $k \in M$.

- x_{ijk} es una variable binaria que vale 1 si el vehículo $k \in M$ hace el recorrido de $i \in V$ a $j \in V$ y 0 en otro caso.

- Q_{ik} es una variable que nos indica la carga que tiene el vehículo $k \in M$ al salir del nodo $i \in V$.

- B_{ik} es una variable que indica el tiempo en el que comienza el servicio del vehículo $k \in M$ en el nodo $i \in V$.

- $\delta^+(i) = \{j \in N/(i,j) \in A\}$ conjunto de arcos de salida y $\delta^-(j) = \{i \in N/(i,j) \in A\}$ conjunto de arcos de entrada.

2.2.2. Objetivo y restricciones

El objetivo para estos problemas se describe como sigue:

$$\sum_{k \in M} \sum_{(i,j) \in A} c_{ijk} x_{ijk}$$

Este resultado vendrá asociado a las siguientes restricciones para el problema que estamos modelizando:

1. Todo nodo que no represente a los nodos de salida y entrada de los vehículos (nodos 0 y $2n+1$) son visitados exactamente una vez:

$$\sum_{k \in M} \sum_{(i,j) \in A} x_{ijk} = 1 \quad \forall i \in P \bigcup D$$

2. Todo vehículo comienza en el nodo de salida:

$$\sum_{j \in \delta^+(0)} x_{0jk} = 1 \quad \forall k \in K$$

3. El final de la ruta de un vehículo acaba en el nodo de finalización:

$$\sum_{i \in \delta^-(2n+1)} x_{i(2n+1)k} = 1 \quad \forall k \in K$$

Estas dos últimas restricciones nos permiten no tener que utilizar todos los vehículos a disposición. Tampoco será necesario. En caso de que no usemos un vehículo, tendremos $x_{0,2n+1,k} = 1$ con $k \in K$.

4. En los nodos que no son almacén, cuando se alcanza un nodo, debe irse con el mismo vehículo:

$$\sum_{i \in \delta^-(j)} x_{ijk} - \sum_{i \in \delta^+(j)} x_{jik} = 0 \quad \forall j \in P \bigcup D \quad \forall k \in K$$

5. Evitamos los subciclos con las unidades de tiempo:

$$x_{ijk} = 1 \Rightarrow B_{jk} \geq B_{ik} + d_i + t_{ijk} \quad \forall (i,j) \in A \quad \forall k \in K$$

6. La carga en un nodo es la carga a la salida del anterior más la carga que hay en ese nodo:

$$x_{ijk} = 1 \Rightarrow Q_{jk} = Q_{ik} + q_j \quad \forall (i,j) \in A \quad \forall k \in K$$

7. Mantenemos la carga entre 0 y la máxima permitida por el vehículo:

$$max\{0, q_i\} \leq Q_{ik} \leq min\{C_k, C_k + q_i\} \quad \forall i \in V \quad \forall k \in K$$

8. Los nodos relacionados de carga y demanda se deben realizar en la misma ruta:

$$\sum_{j \in \delta^+(i)} x_{ijk} - \sum_{j \in \delta^+(n+i)} x_{(n+i)jk} = 0 \quad \forall i \in P \quad \forall k \in K$$

9. La entrega sólo se puede realizar después de haber cargado su respectivo nodo:

$$B_{ik} \leq B_{n+i,k} \quad \forall i \in P \quad \forall k \in K$$

Podríamos tener dos restricciones más, dependiendo de como queremos afrontar el problema:

10. Podríamos tener que cumplir con unas ventanas de tiempo en cada nodo:

$$e_i \leq B_{ik} \leq l_i \quad \forall i \in V \quad \forall k \in K$$

11. Los vehículos podrían tener una duración total de trabajo limitada, por lo cual debemos tener en cuenta las duraciones máximas de cada vehículo:

$$B_{2n+1,k} - B_{0k} \leq T_k \quad \forall k \in K$$

2.3. Problema de transporte de muestras biomédicas (BSTP)

Este problema presenta una red con nodos de recolección y nodos de llegada, siendo una variante del problema antes descrito. Lo aquí expuesto está basado en el artículo de Anaya-Arenas et al. (2019), donde se trata un caso concreto de este problema aplicado para la provincia canadiense de Québec, y en el cual se analiza las recogidas y transportes de muestras biomédicas y las correcciones que se podrían hacer para la selección de laboratorios de llegada desde los distintos lugares de recogida más allá de las propias separaciones de la provincia.

De este problema podemos hallar varias formulaciones diferentes. La que mostraremos será una adaptación del modelo *extended graph* o grafo extendido, pero teniendo en cuenta sólo un elemento de recogida por nodo. Por otro lado, tenemos otras dos formulaciones interesantes de este problema: La ya mencionada *extended graph* y la denominada *multiple transportation requests*.

2.3.1. Conjuntos, parámetros y variables

- $G = (V, A)$ es un grafo donde:
  - $V = \{0, 1, ..., n, n+1\}$ es el conjunto de vértices, de los cuales $N = \{1, ..., n\}$ representan las localizaciones de los clientes, y siendo los valores 0 y $n + 1$ representantes del lugar de salida y llegada de todos los vehículos (el almacén, en nuestro caso, el laboratorio).
  - $A = \{(i, j)/i \neq j, i, j \in V\}$ es el conjunto de arcos que unen los vértices. Asociado a cada uno de estos arcos, tenemos su tiempo de trayecto t_{ij} y distancia d_{ij} de ir desde el nodo $i \in V$ al nodo $j \in V$.
- $M = \{1, ..., m\}$ es el conjunto de vehículos disponibles para la realización de las rutas. Se establece que los costes de los vehículos van a ser iguales sin importar el vehículo usado de este conjunto. Asociado a cada vehículo $k \in M$, tenemos un tiempo límite T_k, que viene dado por el tiempo de trabajo posible para cada vehículo. En general, también habrá un tiempo máximo de transporte para cada muestra, que se representa como T_{max}.
- $R = \{1, ..., l\}$ son las distintas rutas disponibles para cumplir con las demandas. Cada vehículo puede realizar varias de estas rutas, pero teniendo en cuenta siempre el límite de tiempo de trabajo antes descrito.
- Para el cliente $i \in N$, las recogidas deben realizarse en una ventana de tiempo $[a_i, b_i]$, siendo a_i el momento más temprano posible a llegar (si se llega más temprano que este instante, tendrá que esperar) y b_i, el más tarde posible. Estas condiciones se considerarán restricciones duras, es decir, aquellas que deben cumplirse obligatoriamente, sin márgenes de cumplimiento ni penalizaciones en el objetivo.
- τ_i es el tiempo de carga en cada cliente $i \in N$, siendo τ_0 el tiempo de descarga en el laboratorio antes de poder iniciar una nueva ruta.
- x_{ijkr} es una variable binaria que vale 1 si el camino entre $i \in V$ y $j \in V$ se utiliza por el vehículo $k \in M$ en la ruta $r \in R$, y valdrá 0 en cualquier otro caso.

- u_{ikr} es una variable continua que indica el instante de tiempo en el que se realiza la visita al cliente $i \in N$ por el vehículo $k \in M$ en la ruta $r \in R$.
- $\delta^+(i) = \{j \in N/(i,j) \in A\}$ conjunto de arcos de salida y $\delta^-(j) = \{i \in N/(i,j) \in A\}$ conjunto de arcos de entrada.

2.3.2. Objetivo y restricciones

El objetivo de este problema será el siguiente:

$$min \sum_{i=0}^{n} \sum_{j=1}^{n+1} \sum_{k=1}^{m} \sum_{r=1}^{l} d_{ij} x_{ijkr}$$

Las restricciones que se tienen en cuenta en esta formulación son las siguientes:

1. Todos los nodos deben ser visitados:

$$\sum_{k=1}^{m} \sum_{r=1}^{l} \sum_{i \in \delta^-(j)} x_{ijkr} = 1 \quad j \in N$$

2. Cada nodo es visitado como mucho una vez por un vehículo y en una ruta:

$$\sum_{i \in \delta^-(j)} x_{ijkr} \leq 1 \quad j \in N, \quad k \in M, \quad r \in R$$

3. De cada nodo cliente visitado hay que salir:

$$\sum_{i \in \delta^-(j)} x_{ijkr} - \sum_{i \in \delta^+(j)} x_{jikr} = 0 \quad j \in N, \quad k \in M, \quad r \in R$$

4. Posible comienzo de ruta desde el almacén:

$$\sum_{j \in \delta^+(0)} x_{0jkr} \leq 1 \quad k \in M, \quad r \in R$$

5. Toda ruta iniciada debe volver al almacén:

$$\sum_{j \in \delta^+(0)} x_{0jkr} - \sum_{j \in \delta^-(n+1)} x_{j(n+1)kr} = 0 \quad k \in M, \quad r \in R$$

6. Respeto por el orden de las rutas:

$$\sum_{j \in \delta^+(0)} x_{0jkr} - \sum_{j \in \delta^+(0)} x_{0jk(r-1)} \leq 0 \quad k \in M, \quad r \in R \backslash \{1\}$$

7. Estimación de llegada al nodo y eliminación de subciclos:

$$u_{ikr} + \tau_i + t_{ij} - u_{jkr} \leq T_k(1 - x_{ijkr}) \quad i \in \{0, ..., n\} \quad j \in \{1, ..., n+1\} \quad k \in M \quad r \in R$$

8. Restricción de llegadas a tiempo en el intervalo de tiempo especificado en cada nodo:

$$a_j - T_k \left(1 - \sum_{i \in \delta^-(j)} x_{ijkr}\right) \leq u_{jkr} \leq b_j + T_k \left(1 - \sum_{i \in \delta^-(j)} x_{ijkr}\right) \quad j \in N, \quad k \in M, \quad r \in R$$

9. El comienzo de una ruta se establece después del final de la anterior:

$$u_{0kr} \geq u_{n+1,k,r-1} \quad k \in M, \quad r \in R \backslash \{1\}$$

10. Tener en cuenta el tiempo máximo de transporte:

$$u_{n+1kr} - u_{jkr} \leq T_{max} + \left(1 - \sum_{i \in \delta^-(j)} x_{ijkr}\right) T_k \quad j \in N, \quad k \in M, \quad r \in R$$

11. Respeto por el tiempo máximo de trabajo en cada vehículo:

$$u_{n+1kr} - u_{0k1} \leq T_k \quad k \in M, \quad r \in R$$

2.3.3. Otras formulaciones

Como ya se ha comentado, esta formulación sólo tiene en cuenta una recogida de todos los elementos de cada nodo. Sin embargo, las formulaciones *multiple transportation requests* y *extended graph* permiten recogidas en diferentes tiempos, de formas distintas. La primera formulación tiene en cuenta estas recogidas estableciendo una nueva variable binaria y_{jqkr} siendo q la recogida q-ésima posible en el nodo j, y valiendo 1 si este elemento se recoge en este nodo con el camión k en la ruta r y 0 en otro caso, dando unas restricciones parecidas a las que hemos descrito, pero permitiendo la visita a cada nodo varias veces hasta cumplir la demanda de cada uno.

Por otro lado, la *extended graph* o grafo extendido utiliza, como su nombre indica, un grafo donde los nodos son los elementos a recoger. Las distancias siguen siendo las normales entre nodos, y 0 entre los nodos de elementos del mismo cliente. En comparación a la anterior formulación, nos ahorramos las variables de recogida de cada elemento, pero el grafo, de querer dibujarlo, sería muy grande y lioso.

2.4. Otros problemas relacionados

Entre los problemas relacionados con el nuestro encontramos muchas variantes del problema del viajante (de hecho, los otros problemas antes descritos son variaciones del mismo). Entre ellos tenemos el problema de transporte de muestras biomédicas considerado en Yi (2003) y Yücel el al. (2013), donde se aplica esta variante del TSP para el transporte de sangre, en el primer caso, y para la programación de la recogida de las muestras en los laboratorios de análisis, considerando tiempos de procesado, en el segundo. Otro problema es el problema de recolección y envío, CDP (Coltin y Veloso, 2014) siendo un caso particular con un único punto de entrega del problema de recogida y entrega, PDP (Parragh et al. 2008a, 2008b), antes descrito.

Por otro lado, tenemos los árboles de mínimo coste y con ventanas de tiempo, TWMST (Solomon, 1986) que es un problema de búsqueda del árbol de mínimo coste pero incluyendo las ventanas

de tiempo correspondientes, y su variante, el árbol de expansión mínima con capacidades, CMST (Chandy y Lo, 1973), en el cual un nodo central recibe y envía bienes a un grupo de terminales que circulan por arcos de capacidades limitadas, las cuales limitan el flujo de circulación de esos bienes.

Todos estos problemas tienen una característica común: son NP-duros. Esto significa que todo problema de la clase NP se puede reducir a él en tiempo polinomial, siendo un problema de la clase NP aquel en el cual existe un algoritmo que puede verificar cualquier solución a un ejemplo en dicha clase en tiempo polinomial.

2.5. Comparación con nuestro problema

Para finalizar este apartado, veremos qué puntos comparte nuestro problema con respecto a los analizados. En primer lugar, nuestro problema es una variante del problema del viajante. Tenemos un conjunto de localizaciones que deben ser visitadas obligatoriamente antes de volver al punto de partida. La diferencia será el nodo inicial. En el TSP el nodo de partida es la localización de salida y llegada, que es representada por los nodos 0 y $n+1$. En nuestro caso, los nodos de partida podrán ser cualesquiera y solo obligaremos a la llegada de las rutas en un nodo específico. Esto se relaciona con el BSTP, en el cual tenemos varias rutas y obligatoriedad de pasar por los nodos pedidos.

En cuanto a la relación con el problema de recogida y envío, este problema será más parecido a su variante de recolección y envío, con un sólo nodo de llegada. Los fundamentos vistos en el PDP son válidos para nuestro problema. Además, aquí hablamos también de la posibilidad de incluir ventanas de tiempo para los nodos y el límite de tiempo de cada ruta, elementos que serán interesantes en nuestro problema.

Capítulo 3

Modelado matemático del problema

En este capítulo vamos a presentar el modelo del problema de planificación de las rutas a realizar, explicando también las restricciones contempladas. Esta formulación tendrá elementos de los problemas presentados anteriormente, pudiendo considerarse como una variante del problema del viajante. Este modelo lo utilizaremos posteriormente para ver su rendimiento al escribirse con el lenguaje de modelado AMPL (`https://ampl.com/`) y viendo el tiempo que se tardan en resolver ejemplos con distintos parámetros. Para ello, utilizaremos el *solver* GUROBI. (`https://www.gurobi.com/`)

Para la presentación del problema seguiremos la estructura realizada en el capítulo anterior, comenzando con los conjuntos, los parámetros y las variables que tendremos en cuenta para el problema de transporte a modelar. Después, presentamos el objetivo del problema y sus restricciones, las cuales explicaremos tanto formalmente como con una explicación intuitiva de lo que significan. La formulación del problema estará basada en la aparecida en el documento de trabajo de Casas-Méndez y Davila-Pena (2021b).

3.1. Conjuntos, parámetros y variables

- $G = (N, A)$ es un grafo donde:
  - $N = \{1, ..., n\}$ es el conjunto de vértices, en el cual el nodo 1 representa el hospital de Santiago de Compostela (CHUS), y el resto de nodos son los distintos centros en los que recogeremos las muestras de sangre. Asociado a cada nodo, tenemos el tiempo que lleva hacer las operaciones de carga de las muestras, el cual es representado por P_i. n representará la cantidad de centros (incluyendo el CHUS) con los que estaremos trabajando.
  - $A = \{(i, j)/i, j \in N\}$ es el conjunto de arcos que unen los vértices. Asociado a cada uno de estos arcos, tenemos su coste C_{ij}, el cual se interpreta como el coste (en unidad de tiempo) de ir desde el nodo $i \in N$ al nodo $j \in N$. El conjunto de estos costes se establece como una matriz, a la que llamamos C.

- $R = \{1, ..., r\}$ es el conjunto de rutas que queremos realizar, limitadas por el número de ambulancias que tengamos a disposición. Asociado a cada ruta, tenemos un tiempo máximo de duración, que representaremos como δ_v, $\forall v \in R$, expresado en unidades de tiempo que consideremos. r representa la cantidad de rutas con las que trabajaremos.

- $T = \{0, 1, ..., f-1\}$ es el conjunto de las unidades de tiempo que consideraremos necesarias para realizar la planificación. En nuestro problema, tomaremos los periodos de tiempo de 10 en 10 minutos, representando el periodo 0 cualquier momento entre los 0 y los 10 primeros minutos, el 1 los siguientes 10, es decir, entre los 10 y los 20, etc. La cantidad de unidades de tiempo será establecida a partir de los límites de tiempo fijados anteriormente con los parámetros δ_v y la cantidad de rutas que se realicen.

- X_{ijtv} es una variable binaria que vale 1 si en la ruta $v \in R$ se llega al centro $i \in V$ en el periodo de tiempo $t \in T$ y después se va al centro $j \in V$ y 0 en otro caso. Por lo tanto, la cantidad de variables que tendremos en cuenta con esta formulación será de n^2rf, con $n = card(N)$, $r = card(R)$ y $f = card(T)$.

3.2. Objetivo y restricciones del modelo

Una vez presentados los conjuntos, parámetros y variables, pasamos a la explicación de los restantes elementos del modelado del problema. Comenzamos indicando nuestro objetivo, el cual es minimizar el tiempo total del transporte realizado por las ambulancias.

$$\textbf{Objetivo: } min \sum_{(i,j)\in A, t\in T, v\in R} (C_{ij} + P_i) X_{ijtv}$$

Con respecto a los posibles valores de las variables, tenemos unas restricciones que poner para la construcción de las rutas de acuerdo con las características del problema considerado. La primera es la obligatoriedad de que, en caso de dibujar un grafo, al tener la solución final, ésta tenga una estructura de árbol. Esto es debido a que cada ruta parte de un centro distinto del CHUS y, tras visitar otros centros, terminará en el CHUS (1 restricción).

$$\textbf{Restricción 1: } \sum_{(i,j)\in A, t\in T, v\in R} X_{ijtv} = n - 1$$

Nuestro objetivo es pasar por todos los nodos distintos del primero una vez, por lo que, al ser n nodos, la construcción del árbol necesitará un total de $n-1$ arcos. El árbol resultante en este problema tiene la peculiaridad de representar un total de r rutas. El nodo 1 (CHUS) va a ser aquel en el que acaben todas las rutas, es decir, va a ser necesario que todas las rutas pasen por él. Asociado a esto, tendremos un conjunto de restricciones que asegure esta condición (r restricciones).

$$\textbf{Restricción 2: } \sum_{i\in N\setminus\{1\}, t\in T} X_{i1tv} = 1 \ \forall v \in R$$

Por otro lado, va a ser necesario un conjunto de restricciones que evite que se produzcan ciclos, es decir, si en una ruta se recogen las muestras de un centro, no podremos volver a pasar por ese mismo centro ni en esa ruta ni en otra distinta ($n^3r^2\sum_{t\in T}(f-t)$ restricciones).

Restricción 3: $X_{ijkv} + X_{litu} \leq 1\ \forall i,j,l \in N,\ \forall v,u \in R,\ \forall t,k \in T/t \geq k$

Como ya hemos comentado y modelizado antes, el nodo 1 (CHUS) se alcanzará tantas veces como rutas haya. Sin embargo, al resto de nodos se llegará, a lo sumo, una vez. De hecho, esto pasará para todos los nodos excepto los que inician las rutas, los cuales no provendrán de ningún nodo de la red. ($n-1$ restricciones).

$$\textbf{Restricción 4:} \sum_{i\in N, t\in T, v\in R} X_{ijtv} \leq 1\ \forall j \in N\backslash\{1\}$$

Otra cosa que debemos tener en cuenta es que no tiene sentido ir de un nodo en sí mismo (nfr restricciones).

Restricción 5: $X_{iitv} = 0\ \forall i \in N,\ \forall t \in T,\ \forall v \in R$

Esta restricción tiene sentido incluirla. El hecho de ir de una localización a sí misma no ocupa tiempo, por lo que lo lógico sería considerarlo como solución en caso de hacerlo factible, aunque el resto de restricciones podrían no cumplirse. Por otro lado, todo nodo que no se corresponda al CHUS debe ser antecedente de un nodo distinto, y solamente uno ($n-1$ restricciones).

$$\textbf{Restricción 6:} \sum_{j\in N, t\in T, v\in R} X_{ijtv} = 1\ \forall i \in N\backslash\{1\}$$

Un centro que tiene su recogida en una ruta no debe ser visitado en otra ruta. En consecuencia, debemos impedir que el modelado del problema incluya esta situación
($n(n-1)^2r(r-1)f^2$ restricciones).

Restricción 7: $X_{ijtv} + X_{jlku} \leq 1\ \forall j \in N,\ \forall i,l \in N\backslash\{j\},\ \forall v \in R,\ \forall u \in R\backslash\{v\}\ \forall t,k \in T$

Las restricciones anteriores permiten un diseño conveniente de las rutas con respecto al orden de visita de los distintos centros. Lo que nos queda por afinar son los periodos de tiempo en los que alcanzamos cada nodo. Esto lo conseguiremos con las siguientes dos restricciones.

Restricción 8: $X_{ijkv} + X_{jltu} \leq 1$
$\forall i,j,l \in N,\ \forall v,u \in R,\ \forall t \in T,\ \forall k \in T\backslash\{t\}/t < k + P_i + C_{ij}$
Restricción 9: $X_{ijkv} + X_{jltu} \leq 1$
$\forall i,j,l \in N,\ \forall v,u \in R,\ \forall t \in T,\ \forall k \in T\backslash\{t\}/t > k + P_i + C_{ij}$

Estas dos restricciones nos limitan los periodos de tiempo en los que vamos a acceder a un nodo i. La restricción 8 (más de $n^3r^2\sum_{t\in T}(t+1)$ restricciones) nos indica que el instante de llegada a un nodo no es posible que se produzca antes de que se llegue al nodo anterior, se recojan las muestras correspondientes y se transporten hasta el nodo correspondiente, transcurriendo el tiempo de viaje necesario. Por ejemplo, si alcanzamos el nodo $i \in N$ en el periodo 5 no tiene sentido que el nodo

al que se dirige se alcance en el tiempo 4 o en el tiempo 6 si el transporte con la carga ocupa 2 unidades de tiempo.

Análogamente, la restricción 9 (hasta $n^3r^2\sum_{t\in T}(t+1)$ restricciones) establece que la llegada a un nodo no puede producirse después de la llegada al nodo anterior, la recogida de las muestras en este y el tiempo de viaje entre ambos. Volviendo al ejemplo anterior, lo que hacemos con esta restricción es que, si llegamos a un nodo en periodo 5 y el tiempo de carga y transporte es 2, lleguemos al siguiente nodo en 7, sin tardar más tiempo del estipulado.

Otra característica que queremos para nuestras rutas es que comiencen pronto. Si llamamos $t' \in T$ al periodo de tiempo máximo en el que queremos que comiencen las rutas, entonces tenemos que forzar al comienzo tempranero de las rutas (r restricciones).

$$\textbf{Restricción 10:} \sum_{i\in N, j\in N\setminus\{i\}, t\in[1,t']} X_{ijtv} \geq 1 \ \forall v \in R$$

Nótese que la restricción permite, como es lógico, que haya más de un arco que se pueda realizar en esa ruta y antes del periodo t'.

Para terminar, tenemos dos restricciones más. La primera de ellas es la restricción de llegada de cada vehículo. Como ya se ha indicado anteriormente, para poder afrontar sin esperas la llegada de las muestras al CHUS, lo ideal es que los vehículos llegasen en intervalos de 10 minutos. Este tiempo se establece teniendo en cuenta el número de personas que atienden la recepción de las muestras en el CHUS, así como su tasa de servicio, el número de muestras que se transportan en cada ruta y el objetivo de evitar solapamientos en la llegada, recogida y clasificación de las muestras en el momento de la recepción de las muestras de una ruta con respecto a las que han llegado previamente. Podríamos indicar esta restricción de varias maneras, pero elegiremos la siguiente ($n(n-1)fr(r-1)$ restricciones):

$$\textbf{Restricción 11:} \ X_{i1tv} + X_{j1ku} \leq 1$$
$$\forall i \in N, \forall j \in N\setminus\{i\}, \forall v \in R, \forall u \in R\setminus\{v\}, \forall t \in T,$$
$$\forall k \in T/t + C_{i1} - C_{j1} - 0{,}5 \leq k \leq t + C_{i1} - C_{j1} + 0{,}5$$

La última restricción será la de tiempo límite, por la cual el tiempo de la ruta no debe exceder el valor considerado δ_v, $v \in R$ (r restricciones).

$$\textbf{Restricción 12:} \sum_{i\in N\setminus\{1\}, j\in N, t\in T} (C_{ij} + P_i)X_{ijtv} \leq \delta_v \ \forall v \in R$$

3.3. Consideraciones acerca de la complejidad computacional del modelo

En el problema de recogida de muestras considerado, la red del problema, es decir, los nodos y los arcos que los conectan, así como las distancias entre los nodos, viene dada. Como se ha explicado previamente, estos nodos representan hospitales y centros del área sanitaria de Santiago de Compostela. Con respecto al número de rutas, esta cantidad viene dada por el número de ambulancias disponibles. Otro elemento relevante es el conjunto de tiempos dentro del cual se

van a planificar las rutas. La restricción 12 indica que este conjunto debe ser establecido a partir de los parámetros δ_v, con $v \in R$, que señalan el tiempo máximo de cada ruta. En general, este tiempo se tomará igual a 2 horas, aunque esto se expresará en una unidad de tiempo que será de 10 minutos, valor que nos permite una flexibilidad adecuada para confeccionar las rutas. Por otro lado, la restricción 11, que expresa el deseo de no solapamiento en la llegada y recepción de muestras, nos indica que este tiempo de 2 horas deberá ser incrementado en 10 minutos por cada ruta extra, sin contar la primera puesto que, si sólo se hiciera una ruta, no habría problemas de solapamiento.

Al presentar las variables y restricciones en las secciones anteriores se ha señalado que el número de las primeras es de n^2fr mientras que el de las segundas es del orden de $n^3f^2r^2$ si sólo consideramos una ruta y aumentando hasta el doble con más rutas, lo que provoca la aparición de una gran cantidad de restricciones. Por ejemplo, tomando los valores $n = 5$ y $r = 1$, vamos a tener como mínimo δ_1 periodos de tiempo si es un número entero. Estableciendo este límite de tiempo en la ruta de 12 periodos de tiempo, tendremos que tomar un valor de $f = 12$, por lo que obtendremos 300 variables y 18000 restricciones aproximadamente. A pesar de este elevado número, la opción *presolve* de GUROBI consigue eliminar una buena cantidad tanto de variables como de restricciones, lo que rebaja parcialmente la dificultad computacional de la solución.

Capítulo 4

Resolución con el software AMPL

En este capítulo vamos a analizar el rendimiento de nuestra formulación en el software AMPL, tomando distintas cantidades de nodos, distintas cantidades de rutas y tomando los periodos de tiempo (conjunto T) acorde a las condiciones explicadas en el capítulo anterior ($T = \{0, 1, ..., max\{\delta_v / v \in R\}, ..., max\{\delta_v / v \in R\} + r - 1\}$). Además, los conjuntos de nodos, así como los tiempos de desplazamiento y carga, se obtendrán a partir del conjunto de todas las distancias (redondeadas a 5 minutos o 0,5 periodos de tiempo) de todos los centros con los que vamos a trabajar, separados en rutas y en los días en las que las rutas actuales se realizan, con ayuda del software R (`https://cran.r-project.org/`). Para hacer este análisis usaremos el servidor NEOS (`https://neos-server.org/neos/`) por poder dar mayor estabilidad a los tiempos de cálculo con respecto al uso del ordenador personal. Este servidor nos permitirá un uso de memoria de hasta 3 GB y un tiempo de cálculo máximo de 8 horas. El *solver* a utilizar para este caso será Gurobi. Antes de este análisis, y como consecuencia de determinados problemas observados y los cuales explicaremos, procederemos a retocar el código para su uso en AMPL.

4.1. Problemas con decimales en los costes

De la manera en la que hemos establecido los periodos de tiempo del conjunto T, siendo que cada unidad representa un periodo de tiempo de 10 minutos, vamos a tener, inevitablemente, periodos de tiempo que no sean enteros. Podríamos ajustar estos periodos de tiempo a nuestras necesidades y que $t \in T$ representase periodos de tiempo de 5 minutos. Esto solo sería una solución en este caso, pues con tiempos no redondeados volveríamos a tener este problema. Además, tendríamos otro problema: el aumento tanto de variables (pasamos de tener f elementos en T a tener $2f$ lo que dobla las variables) como de restricciones (son en torno a $n^3 f^2 r^2$ con $r = 1$ y $2n^3 f^2 r^2$ con $r > 1$, por lo que se multiplican por 4 al bajar de 10 a 5 minutos estos periodos).

Este problema en AMPL con la formulación antes presentada sólo va a coger aquellos valores $C_{ij} + P_i$ que sean enteros exceptuando los arcos de unión con el CHUS, lo que podría no darnos la solución realmente mínima o incluso no darnos una solución válida. Para ello, podemos modificar las restricciones que más problemas nos pueden dar con los costes de tiempo decimales: la **Restricción**

8 y la **Restricción 9**. Tal y como están definidas, estas restricciones no consideran ningún periodo de tiempo posible para transportes entre dos nodos cuyo tiempo total (es decir, carga y transporte) no sea un múltiplo de 10, lo que se traduce con nuestros datos en que ese tiempo debe ser un valor entero para que se considere posible para la solución.

Para solucionar esto tendremos que hacer algo con estos valores decimales.

4.1.1. Posible solución 1: Función redondeo

Como el problema se produce cuando tenemos costes de tiempo con decimales, podría ser una opción quitar los decimales aplicando la función redondeo, en las restricciones anteriormente mencionadas. Así, estas restricciones pasarían a ser las siguientes ($round()$ hará referencia a la función redondeo a la unidad):

$$\textbf{Restricción 8b: } X_{ijkv} + X_{jltu} \leq 1$$
$$\forall i,j,l \in N,\ \forall v,u \in R,\ \forall t \in T,\ \forall k \in T \backslash \{t\} / t < k + round(P_i + C_{ij})$$
$$\textbf{Restricción 9b: } X_{ijkv} + X_{jltu} \leq 1$$
$$\forall i,j,l \in N,\ \forall v,u \in R,\ \forall t \in T,\ \forall k \in T \backslash \{t\} / t > k + round(P_i + C_{ij})$$

Con las restricciones así indicadas, el problema nos da una solución con un coste de tiempo más bajo o igual que sin ellas, teniendo, en caso posible, una solución. El ejemplo más claro de solución válida y distinta a la solución real se tiene con el conjunto de nodos del CHUS (obligatorio) y pasando por Bertamiráns, Calo, Milladoiro, Santa Comba y Valdubra. Para verlo, tenemos la siguiente tabla con las soluciones obtenidas en cada caso, donde cada fila indica la localización de un nodo, el coste de cada arco que conecta este nodo con el que aparece en la siguiente fila (Coste) y el periodo de llegada al nodo de salida del arco (t), e indicando el coste total de cada solución (Total). El nodo que señala la localidad que se enlaza con el CHUS aparecerá sucedido con un guion e indicando el CHUS.

Efecto del redondeo en las soluciones dadas por AMPL					
Con redondeo	Total= 10	t	**Sin redondeo**	Total= 11	t
Santa Comba	Coste=2	1	Valdubra	Coste=2	2
Valdubra	Coste=3,5	3	Santa Comba	Coste=4	4
Bertamiráns	Coste=2	7	Milladoiro	Coste=1	8
Calo	Coste=1	9	Calo	Coste=2	9
Milladoiro-CHUS	Coste=1,5	10	Bertamiráns-CHUS	Coste=2	11

Vemos que con redondeo tenemos las soluciones de menor coste de tiempo. Sin embargo, no vamos a tener los tiempos correctos de los periodos en los que vamos a llegar al nodo correspondiente. Estableciendo como correcto el primer periodo de tiempo en el que hacemos los cálculos (en nuestro caso, establecemos que la ruta se inicia en el periodo $t = 1$), podemos ver como llega un momento en el que los periodos de tiempo solo suman estos redondeos, quedando lejos de los periodos de tiempo reales (a partir del tercer centro, el periodo de llegada debería indicar el 6 y no el 7, y sigue arrastrando esta diferencia a partir de ahí). La explicación para esto está en que nuestros tiempos decimales son 0,5, con lo cual esta función siempre va a darnos el siguiente entero, provocando que la suma en los periodos de tiempo siempre vaya a una unidad más de lo que debería. En consecuencia, Restricción 8b y Restricción 9b son eficaces a la hora de darnos rutas de coste mínimo, pero no nos indican los periodos de tiempo correctos. Esto podríamos corregirlo de manera manual, pero sería interesante conseguir las variables adecuadas a partir del modelo del problema.

4.1.2. Posible solución 2: Eliminar los decimales

Por eliminar los decimales no nos referimos, como ya se ha explicado antes, a aumentar el conjunto T para que los tiempos sean enteros, puesto que el aumento en las variables y las restricciones nos aumentaría mucho el tamaño y el tiempo computacional. A lo que nos referimos es a modificar estas restricciones de manera que, sin llegar a usar la función redondeo, no tenga en cuenta los decimales y pueda darnos unos periodos más correctos.

Ahora tendremos que tener en cuenta estas restricciones ($\lfloor\ \rfloor$ indica la función suelo y $\lceil\ \rceil$, la función techo):

Restricción 8c: $X_{ijkv} + X_{jltu} \leq 1$

$\forall i, j, l \in N,\ \forall v, u \in R,\ \forall t \in T,\ \forall k \in T \backslash \{t\} / t < k + \lfloor P_i + C_{ij} \rfloor$

Restricción 9c: $X_{ijkv} + X_{jltu} \leq 1$

$\forall i, j, l \in N,\ \forall v, u \in R,\ \forall t \in T,\ \forall k \in T \backslash \{t\} / t > k + 2(P_i + C_{ij}) - \lceil P_i + C_{ij} \rceil$

Eliminación de decimales ruta Santa Comba		
Quitando decimales	Total= 10	t
Santa Comba	Coste=2	1
Valdubra	Coste=3,5	3
Bertamiráns	Coste=2	6
Calo	Coste=1	8
Milladoiro-CHUS	Coste=1,5	9

En el ejemplo presentado en la tabla anterior los periodos de tiempo están calculados de forma correcta, pero podemos tener un problema con otros conjuntos de datos. Si volviésemos a tener un tiempo con el cual la suma de los decimales llegue o pase de 1, entonces los tiempos pasan a estar mal. Por lo tanto, tenemos un nuevo problema para solucionar: las sumas de tiempos con decimales. Esto lo podemos ver reflejado si tomamos la actual ruta 5 del lunes, con los centros de salud de Carenal, Frades, Galeras, Ordes, Oroso y Vite, además del propio CHUS. Los periodos de tiempo que conforman la solución dada por AMPL con esta configuración en este caso son los dados en la siguiente tabla:

Eliminación de decimales ruta Frades		
Quitando decimales	Total= 11	t
Frades	Coste=2,5	2
Ordes	Coste=2	4
Oroso	Coste=2,5	6
Vite	Coste=1	8
Galeras	Coste=1,5	9
Carenal-CHUS	Coste=1,5	10

Podemos observar que sucede lo comentado. Los periodos de tiempo son correctos hasta que los costes de tiempo decimales llegan a sumar 1, donde ya falla. El periodo de tiempo en el que se llega a Vite debería ser el $t = 9$, pues empezamos la ruta con $t = 2$ y los costes anteriores suman 7 unidades de tiempo. Sin embargo, tenemos señalado $t = 8$, debido a esto. De hecho, esto pasará cada vez que la parte decimal de la suma de los costes llegue a una nueva unidad, aumentando aún más el fallo del periodo de tiempo que debería indicarse.

4.1.3. Solución posterior: Corrigiendo con el software R

Una vez propuestas variaciones de las restricciones 8 y 9 para corregir los problemas con los decimales, lo cual hemos conseguido de forma parcial, podemos pensar en dos opciones: o bien seguimos buscando formas de afinar estos tiempos con AMPL, o bien buscamos ayuda externa. Como ya hemos visto al presentar el problema, plantearlo en AMPL con las herramientas disponibles va a resultar en algo bastante limitado por la cantidad de restricciones necesarias. En consecuencia, podría interesarnos no perder mucho más tiempo en esto y arreglar los periodos de tiempo reales de otra manera. Para nuestro caso, usaremos el software de estadística R.

Lo primero que necesitaremos para obtener las rutas correspondientes será resolver este problema con AMPL. Además, con las restricciones 8 y 9 en su versión c, antes presentadas, los trayectos entre localidades estarán ordenados. Este hecho nos permite mantener la estructura del problema original en AMPL, pues podríamos plantear el mismo problema sin tener en cuenta los periodos de tiempo, lo cual nos daría menos variables y restricciones, eliminando sobre todo aquellas que tratan de ajustar esta magnitud. Sin embargo, el hecho de que estos periodos de tiempo ordenen los distintos arcos del grafo nos va a resultar muy cómodo a la hora de llevar nuestra solución a R y ordenar las rutas.

Al resolver con AMPL y tener activada la opción de omitir ceros (*omit_zero_rows 1*) la solución vendrá dada como una matriz de $n-1$ filas (restricción 1) y cinco columnas. Estas columnas representarán:

- **Columna 1:** Nodo de carga de muestras. Origen del trayecto.
- **Columna 2:** Nodo de destino desde el nodo de carga.
- **Columna 3:** Periodo de tiempo en el que, según la estructura en AMPL, se alcanza el nodo de carga de este arco. Esta columna es la que nos importa para corregir, y la que cambiaremos para que nos dé los periodos de tiempo reales en base a nuestros costes.
- **Columna 4:** Ruta en la que se realiza el trayecto. Lo importante no es el orden, si no que no tengamos nodos distintos del CHUS que se encuentren en varias rutas.
- **Columna 5:** Columna de valores de la variable. Todas indicarán 1 en un inicio. Esto es de poco interés, pues al ser X una variable binaria y tener activada la opción antes descrita, pues se puede considerar que los valores de las variables que aparecen serán 1 sin indicarlo implícitamente. En esta columna pondremos los costes de los arcos, un dato que nos será de más interés.

Para comenzar el proceso de cambio de los periodos de tiempo, lo primero que haremos será cargar la solución en el software R, cargando también la matriz de distancias entre todas las localidades y sumando los tiempos de carga necesarios. A continuación, hacemos el primer ajuste de los periodos de tiempo. Por comodidad, consideraremos que el periodo de tiempo más bajo sea el 0. Así, pues, simplemente restando el mínimo valor de esta columna a todos los elementos tendremos este objetivo cumplido.

Lo siguiente será ordenar los arcos. Para ello, colocaremos los arcos de la misma ruta todos juntos y los ordenaremos de menor a mayor valor de t. Nótese que, con esta configuración, estos tiempos están bien ordenados dentro de cada ruta, como ya hemos comentado. El único problema es que no necesariamente son correctos partiendo del periodo de inicio de la ruta, que es lo que nos lleva a este cambio. Así, pues, nos quedaría una ordenación correcta de como se ordena la llegada a los nodos.

Para verlo mejor se puede volver a las soluciones obtenidas para la sección anterior y poniendo las columnas de la forma en la que tenemos la solución en AMPL. También habremos puesto como $t = 0$ al periodo de tiempo del arco que inicia la ruta (recordar que este periodo establece los

tiempos desde 0 minutos, incluido, hasta 10 minutos, no incluido) y el nombre de las localidades en vez del número de nodo que representan. Todo esto será presentado en las siguientes dos tablas.

Lunes, ruta 3, AMPL			
Arcos	t	Ruta	Coste arco
(Santa Comba, Valdubra)	0	1	2
(Valdubra, Bertamiráns)	2	1	3,5
(Bertamiráns, Calo)	5	1	2
(Calo, Milladoiro)	7	1	1
(Milladoiro, CHUS)	8	1	1,5

Lunes, ruta 5, AMPL			
Arcos	t	Ruta	Coste arco
(Frades, Ordes)	0	1	2,5
(Ordes, Oroso)	2	1	2
(Oroso, Vite)	4	1	2,5
(Vite, Galeras)	6	1	1
(Galeras, Carenal)	7	1	1,5
(Carenal, CHUS)	8	1	1,5

El siguiente paso es corregir los periodos de tiempo y hacer que indiquen el periodo real, es decir, que el tiempo real pueda estar en el periodo de 10 minutos que representa. Lo haremos de manera muy fácil. Partiendo del tiempo inicial de la ruta, el cual representa el periodo de tiempo en el que se llega al centro de salida, y el tiempo de coste de ese arco (incluyendo la carga), tendremos el tiempo para el siguiente arco y así hasta llegar al CHUS. Por último, con aplicar la función suelo a las sumas anteriores ya tendremos los periodos de tiempo reales que deben acompañar a cada arco.

Por lo tanto, las rutas que hemos comprobado antes ahora quedarán con los periodos de tiempo que aparecen en las próximas tablas:

Lunes, ruta 3, corrección R			
Arcos	t	Ruta	Coste arco
(Santa Comba, Valdubra)	0	1	2
(Valdubra, Bertamiráns)	2	1	3,5
(Bertamiráns, Calo)	5	1	2
(Calo, Milladoiro)	7	1	1
(Milladoiro, CHUS)	8	1	1,5

Lunes, ruta 5, corrección R			
Arcos	t	Ruta	Coste arco
(Frades, Ordes)	0	1	2,5
(Ordes, Oroso)	2	1	2
(Oroso, Vite)	4	1	2,5
(Vite, Galeras)	7	1	1
(Galeras, Carenal)	8	1	1,5
(Carenal, CHUS)	9	1	1,5

Al solo tener una ruta resulta ser algo así de fácil. Sin embargo, al tener varias rutas, la restricción 11 establece que haya una separación de 10 minutos entre las llegadas al CHUS. Si tomamos las rutas que acabamos de ver y las combinamos, los resultados obtenidos en tiempos serán los siguientes:

Rutas 3 y 5, lunes, primera corrección			
Arcos	t	Ruta	Coste arco
(Frades, Ordes)	1	1 (5)	2,5
(Ordes, Oroso)	3	1 (5)	2
(Oroso, Vite)	5	1 (5)	2,5
(Vite, Galeras)	8	1 (5)	1
(Galeras, Carenal)	9	1 (5)	1,5
(Carenal, CHUS)	10	1 (5)	1,5
(Valdubra, Santa Comba)	0	2 (3)	2
(Santa Comba, Bertamiráns)	2	2 (3)	3,5
(Bertamiráns, Calo)	5	2 (3)	2
(Calo, Milladoiro)	7	2 (3)	1
(Milladoiro, CHUS)	8	2 (3)	1,5

En este caso, los tiempos de las rutas son de 11 unidades de tiempo en la ruta 5, que es la que está marcada como ruta 1 en la tabla y 10 para la ruta 3. Las rutas resultantes tienen el mismo coste que obtuvimos al hacerlas por separado, y son casi las mismas. Para el ajuste en este caso va a ser suficiente cambiar los tiempos de ruta de la primera para cumplir con la condición de intervalo de 10 minutos entre llegadas al CHUS. Para ello, empezaríamos la ruta con $t = 0$ y el resto irán corrigiéndose para cumplir con los costes del arco correspondiente, quedando nuestra solución de la siguiente manera:

Rutas 3 y 5, lunes, corrección de llegada			
Arcos	t	Ruta	Coste arco
(Frades, Ordes)	0	1 (5)	2,5
(Ordes, Oroso)	2	1 (5)	2
(Oroso, Vite)	4	1 (5)	2,5
(Vite, Galeras)	7	1 (5)	1
(Galeras, Carenal)	8	1 (5)	1,5
(Carenal, CHUS)	9	1 (5)	1,5
(Valdubra, Santa Comba)	0	2 (3)	2
(Santa Comba, Bertamiráns)	2	2 (3)	3,5
(Bertamiráns, Calo)	5	2 (3)	2
(Calo, Milladoiro)	7	2 (3)	1
(Milladoiro, CHUS)	8	2 (3)	1,5

Sin embargo, no siempre va a resultar tan sencillo, por lo que conviene tener un método para corregir estos tiempos y cumplan con la restricción 11. Por ejemplo, si hacemos lo mismo con las actuales rutas 1 y 5 del lunes, los tiempos de cada ruta serán de 10,5 y 11 respectivamente. Sin embargo, el tiempo de inicio después de la primera corrección en cada ruta es de 0 y 2 respectivamente, por lo que los periodos de tiempo de llegada serán $t = 10$ y $t = 13$, como podemos apreciar en la siguiente tabla:

Rutas 1 y 5, lunes, primera corrección			
Arcos	t	Ruta	Coste arco
(Frades, Ordes)	2	1 (5)	2,5
(Ordes, Oroso)	4	1 (5)	2
(Oroso, Vite)	6	1 (5)	2,5
(Vite, Galeras)	9	1 (5)	1
(Galeras, Carenal)	10	1 (5)	1,5
(Carenal, CHUS)	11	1 (5)	1,5
(Muros, Esteiro)	0	2 (1)	2
(Esteiro, Outes)	2	2 (1)	2
(Outes, Noia)	4	2 (1)	2
(Noia, Lousame)	6	2 (1)	1
(Lousame, CHUS)	7	2 (1)	3,5

Como consecuencia de estos problemas, tendremos que pensar una manera de solucionar esto. Lo que haremos será ordenar las rutas por los tiempos de llegada. En este caso, tendríamos la ruta 1 con $t = 10$ antes de la ruta 5 con $t = 13$. Estos son los periodos de tiempo en los que llegarán ahora mismo. Para evitar confusiones en la llegada al CHUS y ayudar a asegurar los diez minutos de intervalo, buscaremos que estas sumas, que indicarán la llegada de cada ambulancia sean enteros y nos indiquen el tiempo justo al inicio de este periodo. Así, comenzaremos transformando el periodo de tiempo de llegada al CHUS a un valor entero, que será el siguiente al valor de la suma de este periodo inicial y la suma de la ruta. En nuestro caso, la ruta 1, de llegada 10,5, pasará a considerarse un valor de $t = 11$, teniendo un poco de margen.

A partir de aquí, los tiempos de llegada irán aumentando en una unidad de tiempo en la siguiente ruta. Por tanto, el periodo de llegada al CHUS para la ruta 5 la estableceremos en $t = 12$. Si tuviesemos una ruta más, indicamos que acabe en $t = 13$, etc. Una vez hecho esto, pasamos a calcular el nuevo periodo de tiempo en el que comenzará la ruta. Para ello, tomamos estos tiempos y les restamos la duración total de cada una de sus rutas. Esto nos puede generar un problema y es que tengamos tiempos de inicio negativos. Esto lo corregimos de manera sencilla: Tomamos el

mayor entero menor al mínimo valor después de la resta y sumamos su valor absoluto a todos los tiempos. Esto nos dará tiempos de inicio positivos. A partir de aquí, sumamos los costes como ya hicimos para la primera corrección y aplicamos la función suelo para tener los distintos periodos de tiempo.

Veamos en que se traduce el hacer estos cambios en nuestros ejemplos. Como ya hemos indicado con las rutas 3 y 5 del lunes, este proceso simplemente lleva ambos tiempos de inicio a $t = 0$ para acabar llegando al CHUS en los periodos $t = 10$ y $t = 11$ respectivamente. Para las rutas más recientes, ya sabemos que los tiempos finales esperados son $t = 11$ y $t = 12$ para las rutas 1 y 5 del lunes. Restando ahora los costes, nos queda que los tiempos de inicio son de $t = 0{,}5$ y $t = 1$ respectivamente. Ese decimal se tendrá en cuenta a la hora de calcular los periodos de tiempo de llegada a los siguientes nodos. Además, ninguno es negativo, por lo que no será necesario hacer una segunda corrección a estos tiempos iniciales. La solución final para este par de rutas juntas aparece en la siguiente tabla:

Rutas 1 y 5, lunes, primera corrección			
Arcos	t	Ruta	Coste arco
(Frades, Ordes)	1	1 (5)	2,5
(Ordes, Oroso)	3	1 (5)	2
(Oroso, Vite)	5	1 (5)	2,5
(Vite, Galeras)	8	1 (5)	1
(Galeras, Carenal)	9	1 (5)	1,5
(Carenal, CHUS)	10	1 (5)	1,5
(Muros, Esteiro)	0	2 (1)	2
(Esteiro, Outes)	2	2 (1)	2
(Outes, Noia)	4	2 (1)	2
(Noia, Lousame)	6	2 (1)	1
(Lousame, CHUS)	7	2 (1)	3,5

De todas maneras, podríamos tener problemas para cumplir la restricción 10, la cual nos obliga a iniciar las rutas antes de un determinado periodo de tiempo. Para nuestro caso, las salidas deben suceder antes del paso de dos horas, es decir, entre los periodos $t = 0$ y $t = 11$, ambos incluido. Este hecho, unido a la restricción 12 para nuestro problema, en el cual limitamos los tiempos de ruta a 12 unidades de tiempo, nos indica que, a pesar de las correcciones, no debería haber inconveniente en el periodo de tiempo del inicio de ruta. En el caso de que hubiese problemas por esta restricción significaría que el tiempo de alguna ruta no cumple esta última restricción y sería necesaria una revisión de los centros de salud involucrados en el conjunto de rutas a analizar.

Para acabar con esta sección, señalar que este proceso puede realizarse también con soluciones heurísticas, las cuales podrían ocasionar algún problema extra a la hora de preparar los datos, pues varios fundamentos de estas modificaciones se ayudan de la estructura que tenemos en la forma de presentar la solución proveniente de AMPL.

4.2. Rendimiento del problema en AMPL

Veamos qué sucede cuando consideramos distintas situaciones en AMPL para ver su rendimiento. Para ello, veremos el tiempo aproximado de resolución de varios problemas cumpliendo con todas las restricciones. Para tener mayor certeza de su cumplimiento, veremos los casos con las rutas ya conocidas, pues en general cumplirán con los límites de tiempo de rutas estipulados. En caso de que no sea así, nos aparecerá como un problema sin solución, tipo de caso que no tendremos en cuenta para estos análisis.

El problema a resolver es de tipo MILP, es decir, un problema de programación mixta lineal-entera. Para la resolución de este tipo de problemas en AMPL se utilizarán dos tipos de algoritmos: el método símplex y el branch and bound (B&B) o de ramificación y poda. La necesidad de realizar iteraciones en ambos métodos será lo que nos vaya a dar el rendimiento final de resolución del problema.

El razonamiento lógico para los tiempos de resolución sería que, a más variables y restricciones sea necesario tener en cuenta, mayor número de iteraciones necesitaremos en los métodos que resuelven el problema para llegar a una solución óptima, lo que resultaría en una mayor cantidad de tiempo. En base a este razonamiento, buscamos ver cual sería el límite de elementos del problema con los cuales podríamos obtener una solución en un tiempo de resolución lógico, que estableceremos en una hora. Sin embargo, es cierto que, en caso de ser posible no nos resultaría un problema el hecho del tiempo de resolución para unas pocas ejecuciones del problema.

Comenzaremos con el caso de una ruta. En este caso, tendremos una ruta ($r = 1$) y doce periodos de tiempo a tener en cuenta ($f = 12$) dentro del límite establecido para llegar a las 2 horas del límite que queremos cumplir. En la siguiente tabla aparece el rendimiento de este problema para las distintas cantidades de nodos que, con las rutas ya establecidas, cumplen todas las restricciones. En aquellos casos posibles, utilizaremos varias rutas para extraer la cantidad de restricciones y el tiempo de resolución a partir de la media de estas rutas.

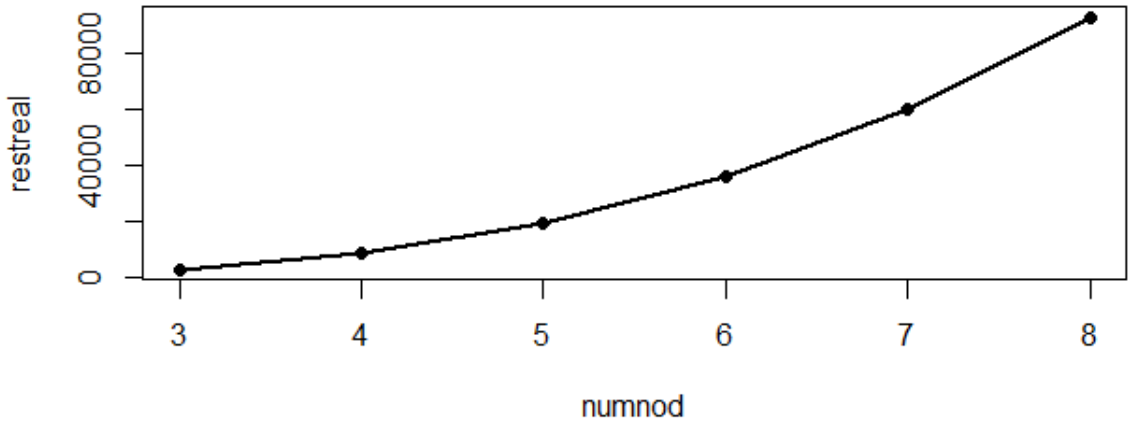
Tabla de rendimiento para cálculos de una ruta				
Nodos	Variables	Restricciones (aprox.)	Restricciones (real)	Tiempo (en s)
3	108	3888	2848	0,043387
4	192	9216	8560	0,067545
5	300	18000	18947	0,1400003
6	432	31104	35552	0,370682
7	588	49392	59562	0,5539437
8	768	73728	92656	0,750205

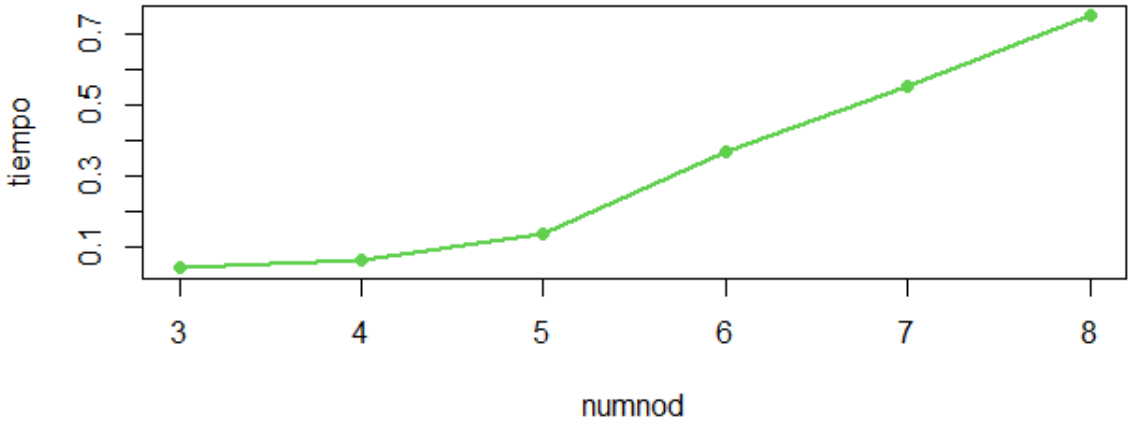
Las columnas de las tablas nos van a indicar, de izquierda a derecha, el número de nodos con el que hemos trabajado, el número de variables que tendremos, basadas en los valores de n, r y f que tengamos, el número de restricciones basado en la aproximación anteriormente comentada ($n^3r^2f^2$ cuando tenemos $r = 1$ y el doble a partir de dos rutas), la media de las cantidades de restricciones reales que hemos tenido en los distintos problemas planteados con la misma cantidad de rutas y nodos y el tiempo medio en el que se han resuelto los distintos problemas con la misma cantidad de nodos y rutas.

Los resultados obtenidos son los que cabía esperar: A medida que aumentamos los nodos, aumentan las variables y las restricciones y esta situación también provoca un aumento del tiempo de resolución, el cual en todo momento es casi instantáneo sin llegar a tardar un segundo.

En cuanto a la comparación del número de restricciones aproximado por nosotros y el real, podemos observar como en las dos primeras cantidades hemos predicho más de las que han sido realmente, mientras que, a medida que aumentaba el número de nodos considerados, este valor de restricciones era realmente mayor. Sin embargo, podemos considerar que nuestra aproximación rápida de $n^3r^2f^2$ restricciones es buena, sobre todo para hacernos una idea previa de lo que podemos esperar para estos problemas.

A continuación presentamos un par de gráficas que relacionan el número de nodos en el problema con la cantidad de restricciones reales utilizadas en cada caso (en negro) y los nodos con el tiempo de resolución (en verde):





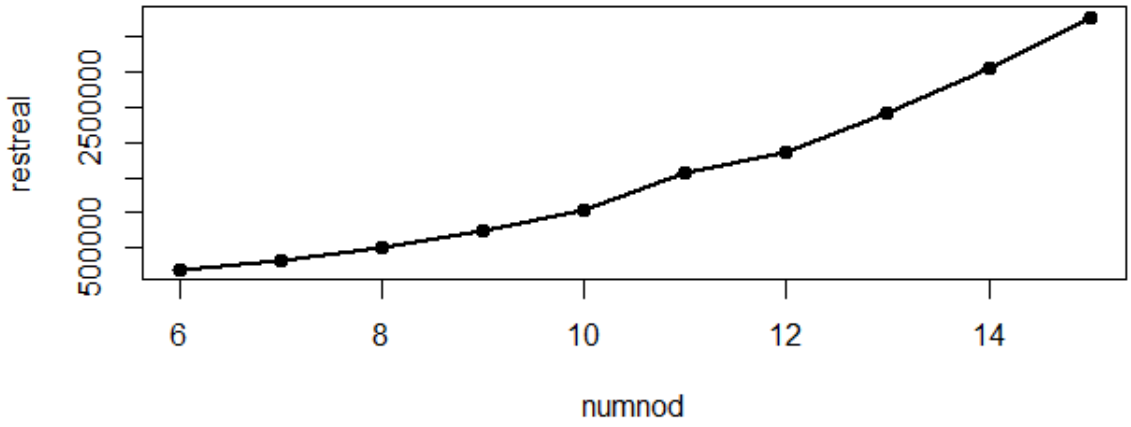
Podemos observar como las gráficas son similares, si bien tenemos alguna variación entre nodos que parece distinta en algún punto cuando vemos la gráfica de los tiempos de resolución de los problemas. Por lo tanto, si bien la cantidad de variables y restricciones que tengamos van a resultar importantes, puede que no sea lo único que nos cambie el tiempo final de resolución.

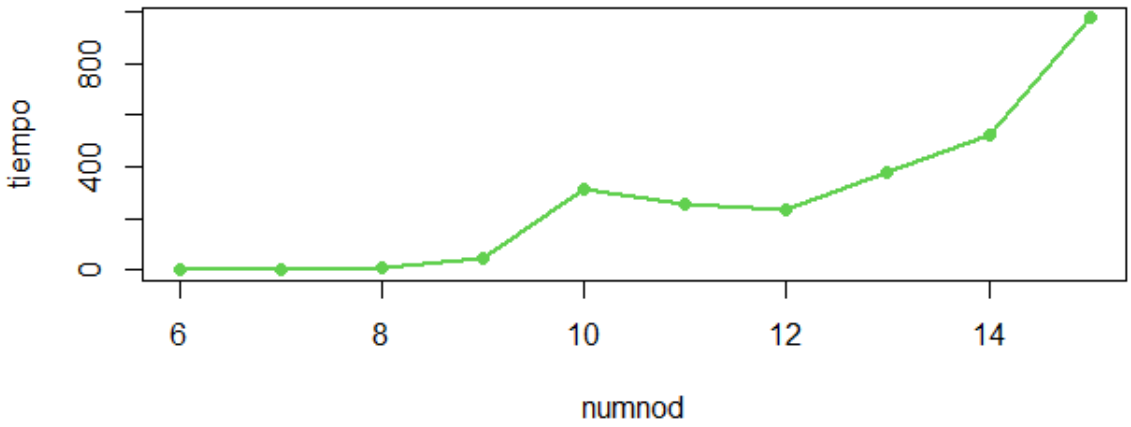
En base a estas observaciones, partimos a analizar qué sucede cuando el problema requiere buscar dos rutas ($r = 2$). Como ya hemos explicado, el hecho de esperar 10 minutos entre la llegada al CHUS de una ambulancia y la siguiente, para no tener saturación en el momento de la recogida de las muestras, obliga a aumentar la cantidad de periodos de tiempo, teniendo en este caso $f = 13$ periodos de tiempo a tener en cuenta. De esta manera, para cada cantidad de nodos de cada caso vamos a tener la siguiente tabla de rendimiento:

Tabla de rendimiento para cálculos de dos rutas combinadas				
Nodos	Variables	Restricciones (aprox.)	Restricciones (real)	Tiempo (en s)
6	936	292032	193822	1,59595
7	1274	463736	324560	3,202963
8	1664	692224	505812	11,7085
9	2106	985608	742968	46,91523
10	2600	1352000	1047354	312,7217
11	3146	1799512	1570706	255,6534
12	3744	2336256	1871840	230,9873
13	4394	2970344	2412937	375,8995
14	5096	3709888	3051512	520,897
15	5850	4563000	3784987	976,517

En este caso tenemos la aproximación de $2n^3f^2r^2$, la cual es bastante buena pero siempre está por encima del número de restricciones que se utilizan en realidad. De todas maneras, estas aproximaciones nos resultan lo suficientemente válidas para analizar si se puede considerar posible una resolución en un tiempo razonable del problema que podamos tener.

Por otro lado, el salto de tiempo es muy grande, pasando de tardar menos de dos segundos con pocos nodos a llegar a pasar los 15 minutos. Además no es progresivo, como sucedía al solo ver una ruta, pues vemos como en la mitad de la tabla vemos algunos tiempos de resolución menores a los obtenidos anteriormente con menos nodos, lo cual podemos observar en la siguiente gráfica:





Mientras la curva de las restricciones podemos decir que tiene un dibujo similar al del caso de una ruta, aunque a otra escala, el carácter no creciente en algunos tramos de la gráfica del tiempo y de lo cual no tendríamos por qué tener una previsión antes de resolver los problemas, dibujan una gráfica que dista del dibujo anterior o del previsible. Esto puede explicarse con la complejidad de los problemas.

Con una ruta es lógico pensar que con el aumento de la cantidad de nodos haya también un aumento de rutas posibles y resulte más complejo llegar a una solución óptima, como ya se ha comentado anteriormente. Este pensamiento debería poder llevarse también de forma general, sin importar la cantidad de rutas que queramos obtener. Sin embargo, podría darse el caso de que la solución del problema resulte más compleja con menos nodos y viceversa, afectando esto en el

tiempo final de resolución. Esto resultaría así por una mayor cantidad de iteraciones de los métodos utilizados para la resolución, sobre todo con el B&B, pues podríamos decir que cada uno de los cortes sería como tener un subproblema nuevo que requiere de realizar más iteraciones del símplex. De todas maneras, esto podríamos considerarlo como algo propio de cada problema que vayamos a resolver. De esta forma podemos, simplemente, partir de un tiempo base de resolución que resulte más alto a mayor cantidad de nodos cuando tenemos los mismos periodos de tiempo y rutas y, por consiguiente, de restricciones, considerando el razonamiento inicial como una simplificación de lo que sucede en general más que como un hecho.

En cuanto a la cantidad de restricciones, vemos como la estimación inicial ya sobrepasa los 4,5 millones de restricciones con el caso de 15 nodos, dando un tiempo de resolución todavía razonable. Cabe decir también que la restricción 11 de límite de tiempo de las rutas ayuda a este tiempo de resolución, pues limita bastante la cantidad de soluciones posibles, lo que hace que se pueda buscar más rápido entre éstas. Si hacemos este mismo problema sin esta restricción el tiempo de resolución llega al límite. En consecuencia, podemos decir que esta restricción ayuda a su resolución al acotar la cantidad de soluciones posibles. Esto podría contradecir nuestra suposición inicial de a mayor cantidad de restricciones, más tiempo, pero podríamos considerar este problema como uno diferente y no entraría dentro de esta lógica el comparar dos problemas que son, en esencia, diferentes.

Nos interesa ver hasta donde podemos abarcar con nuestras herramientas. Es por ello que procedemos a realizar este problema ahora para tres rutas ($r = 3$). Esto nos obliga, nuevamente, a añadir un periodo de tiempo más, pasando ahora a llegar desde $t = 0$ hasta $t = 13$ ($f = 14$). Esto ya nos va a generar una cantidad de restricciones muy grande con pocos nodos con los que vayamos a trabajar. Los resultados de rendimiento posibles aparecen en la siguiente tabla:

Tabla de rendimiento para cálculos de tres rutas combinadas

Nodos	Variables	Restricciones (aprox.)	Restricciones (real)	Tiempo (en s)
10	4200	3528000	2979316	177,371
11	5082	4695768	4054466	181,137
12	6048	6096384	5336468	734,5173

En cuanto al rendimiento seguimos teniendo un comportamiento general similar al obtenido en anteriores casos, con un aumento notable a mayor cantidad de nodos observada. Sin embargo, la cantidad de nodos con la que podemos trabajar en nuestras condiciones es 12. Al querer solucionar un problema con 13 nodos nos encontramos con un mensaje de límite de tamaño permitido, es decir, que el problema resulta demasiado grande como para poder solucionarlo con las condiciones que tenemos. En este caso, llegamos a tener aproximadamente unas 7,75 millones de restricciones, y parece que, para nuestros problemas, llegar a esas cantidades supera los 3GB permitidos por el servidor NEOS con el que estamos trabajando. Por lo tanto, hemos llegado a una cantidad

límite para nuestro problema. Si lo intentamos con otros problemas de las mismas características o con más nodos, resultará improbable obtener una solución por culpa de este límite. Además, cabe añadir que se ha tenido incluida en todo momento la opción que habilita una solución previa (el *presolve*) lo que alivia parte de la carga del problema. Por consiguiente, si intentar resolver un problema con estas características resulta casi imposible con $n = 13$, $r = 3$ y $f = 14$, entonces podemos considerar inviable la resolución del problema con todas las rutas en cada uno de los días.

Seleccionando el lunes, por ejemplo, tendríamos $n = 43$, $r = 7$ y $f = 18$, esto último en el supuesto de que todas las rutas tengan un máximo de dos horas y no tengamos lo que sucede en la realidad, con una ruta que sobrepasa las tres. Con estas condiciones, el número de variables que tenemos es de 232974 y, llegando la cantidad de restricciones a poder estar en torno a los mil millones, podemos ver que afrontar este problema directamente va a resultar muy difícil.

4.3. Conclusiones del uso de AMPL

El uso del software AMPL para la resolución del problema, más allá de la validación del modelo, puede resultar problemático. Desde la presentación de los datos hasta el tamaño del problema y el tiempo que pueda llevar solucionarlo, nos encontramos con ciertas limitaciones que nos obligan a pensar en otras formas de poder resolver nuestro problema dentro de nuestras condiciones con el uso del servidor NEOS.

Es cierto que la modelización del problema en este software es muy sencilla, siendo prácticamente directa. Sin embargo, las restricciones 8 y 9, aquellas que tratan de darnos los periodos de tiempo correctos de llegada a las distintas localizaciones, pueden llegar a complicar el problema. En consecuencia, y en búsqueda de obtener los periodos de tiempo correctos, se han buscado modificaciones de las restricciones que puedan suplir este problema, sin llegar a tener un éxito total. Estos periodos correctos hemos logrado obtenerlos de forma manual, con ayuda de un software externo.

Por otro lado, el rendimiento con la configuración actual nos lleva a tener resultados hasta llegar a unos seis o siete millones de restricciones, con tiempos de resolución bastante manejables. Esto es consecuencia de la limitación de tamaño que tenemos, y con un ordenador más potente podremos llegar a manejar problemas más complejos. De todas maneras, no parece que los tiempos de resolución para los problemas que querríamos abarcar sean muy manejables, lo que también resulta problemático.

Dentro de las posibles soluciones estaría la modificación del problema. Ya hemos visto que la solución más rápida y sencilla para establecer los tiempos de llegada a las distintas localidades es utilizar un software externo y realizarlo de manera independiente a AMPL. Por lo tanto, simplemente se podrían quitar los periodos de tiempo del problema, eliminando también las restricciones asociadas al tiempo, aunque probablemente debamos incluir alguna restricción extra para evitar ciclos. A partir de esta solución, realizamos el proceso ya explicado en este capítulo, teniendo que ordenar antes los arcos de manera adecuada. En el Apéndice B aparecen los archivos del problema para su funcionamiento en AMPL.

En caso de no querer cambiar esto, o sencillamente por dar otro método para solucionar nues-

tro problema, podríamos tratar de buscar una solución por un método heurístico, buscando una forma de resolución propia con ayuda de otro software distinto. Si bien este sistema podría no dar soluciones óptimas, podrían resultar en soluciones aceptables en tiempos de resolución mucho más rápidos que con el software de AMPL. Este tipo de resolución pasamos a analizarlo en el siguiente capítulo.

Capítulo 5

Resolución heurística del problema

Una vez concluido que la resolución directa en AMPL a partir de ciertas dimensiones es inviable, nos tocará buscar otra forma de poder solucionar nuestro problema. La mejor manera para solucionar un problema de este calibre será utilizar un método heurístico. Para aplicarlo, tendremos que partir de una solución inicial e ir modificándola poco a poco, cumpliendo las restricciones de nuestro problema, hasta llegar a una solución que podamos considerar válida, si bien no necesariamente debe ser la óptima. Una solución se considerará válida cuando cumpla con las restricciones del problema y, dentro de las modificaciones que realicemos, no se consiga mejorar la solución ya establecida.

Este tipo de algoritmos heurísticos resulta ser habitual para la resolución de problemas de rutas de vehículos en general y, de manera más particular y cercana a nuestro problema, en problemas de recogida de muestras biológicas. Entre otros estudios y artículos previos, podemos citar a Mobasher et al. (2015), cuyo estudio de las operaciones de recogida y citas en centros de donación de sangre con un tiempo máximo de llegada al laboratorio y con una única recogida en cada lugar contiene una propuesta de un algoritmo heurístico para su resolución. También podemos hablar de Anaya-Arenas el al. (2016), artículo previo al ya comentado en el Capítulo 2 y en el cual se estudia el problema donde cada centro requiere varias visitas con ventanas de tiempo independientes y limitando la duración de los viajes; y Naji-Azimi et al. (2016) considera una variante en la cual se minimiza el número de llegadas de vehículos al laboratorio en cada espacio de tiempo. Por último, cabe dar mención a Kergosien et al. (2013), donde se estudia un problema de rutas para recoger muestras de pruebas médicas en los servicios de asistencia sanitaria a domicilio, proponiendo para su resolución distintas metaheurísticas.

En este trabajo se dará la propuesta de un algoritmo heurístico basado en ideas intuitivas y que pretende, de forma rápida, reducir el coste de un conjunto de soluciones iniciales para las rutas que pretendemos diseñar. Antes de trabajar con el método heurístico, propondremos un método para buscar una solución inicial que no necesite de demasiados cambios para aligerar el tiempo de computación del método heurístico.

Para ejemplificar la búsqueda de solución inicial, así como los distintos cambios que vayamos realizando, utilizaremos las distancias en la unidad de tiempo ya establecida (1 unidad de

tiempo=10 minutos) entre las localidades en las que se realiza actualmente la denominada ruta 1 los lunes, por lo cual estos pasos los aplicaremos aquí para una sola ruta, y siendo dichas localizaciones el CHUS (nodo 1, siempre presente), Esteiro, Lousame, Muros, Noia y Outes (nodos 23, 30, 35, 37 y 42, respectivamente). Los tiempos de transporte y carga entre estas localidades son los siguientes:

$$\begin{pmatrix} 0 & 4 & 3 & 5 & 3.5 & 3 \\ 4.5 & 0.5 & 2.5 & 2 & 2.5 & 2 \\ 3.5 & 2.5 & 0.5 & 4 & 1 & 2 \\ 5.5 & 2 & 4 & 0.5 & 4 & 3.5 \\ 4 & 2.5 & 1 & 4 & 0.5 & 2 \\ 3.5 & 2 & 2 & 3.5 & 2 & 0.5 \end{pmatrix}$$

Nótese que:

1.- Las localidades están en el orden presentado anteriormente.

2-. Estos tiempos incluyen en las filas los tiempos de carga en los distintos centros, sumados a los tiempos de transporte. Por lo tanto, la matriz indicará el tiempo total de carga y transporte entre la fila i y la columna j.

5.1. Búsqueda de solución inicial

Comenzamos el proceso buscando una solución inicial a partir de la cual aplicar criterios para intentar mejorarla. Las restricciones que tomaremos para tomar esta primera solución serán las restricciones de la 1 a la 7, es decir, aquellas que se corresponden a la estructura que queremos que tenga el grafo sin considerar los periodos de tiempo (que podemos tomar posteriormente de forma manual al tener una solución) y sin tener en cuenta restricciones de tiempo. Estas últimas se pueden aplicar posteriormente al realizar cambios. Por consiguiente, ya tenemos un criterio a partir del cual obtener una solución inicial.

Un posible criterio de elección para esta solución inicial podría ser elegir los nodos y, según la numeración que tengan los nodos que corresponden a las distintas localidades consideradas, ir de mayor a menor hasta alcanzar el nodo 1, es decir, el CHUS. En nuestro caso, sería hacer la ruta Outes (42), Noia (37), Muros (35), Lousame (30), Esteiro (23) y CHUS (1). Los costes serán los rodeados de rojo en la siguiente matriz:

$$\begin{pmatrix} 0 & 4 & 3 & 5 & 3.5 & 3 \\ 4.5 & 0.5 & 2.5 & 2 & 2.5 & 2 \\ 3.5 & 2.5 & 0.5 & 4 & 1 & 2 \\ 5.5 & 2 & 4 & 0.5 & 4 & 3.5 \\ 4 & 2.5 & 1 & 4 & 0.5 & 2 \\ 3.5 & 2 & 2 & 3.5 & 2 & 0.5 \end{pmatrix}$$

Se puede observar que las restricciones indicadas anteriormente no se incumplen: Tenemos un arco menos que la cantidad de nodos con la que trabajamos, llegando al CHUS, sin subciclos, siendo los nodos que no representan al CHUS antecedentes de otros y con, a lo sumo, una procedencia anterior. Al ser solo una ruta, no tendremos el problema de intersección de rutas. Con esta solución, tendremos un total de 17 unidades de tiempo, por lo que esta ruta tardará casi 3 horas en realizarse.

Este criterio puede tener problemas. El primero es la elección con varias rutas. Hacer esta elección con una ruta no conlleva un problema mayor que el tiempo que puede llevar su realización. Sin embargo, cuando tenemos varias rutas deberíamos tener en cuenta otras cosas, por lo que la elección de grupos de nodos para hacer las rutas podría ser arbitrario y ocasionar problemas para la resolución. Por otro lado, tenemos el tiempo de ruta, el cual podría ser muy elevado a la hora de no encontrar posibles mejoras, lo cual tampoco sería lo recomendado.

Así, pues, podríamos pensar en otro criterio para seleccionar las rutas iniciales y que no se base tanto en aleatoriedad. Nuestro objetivo es minimizar el tiempo de transporte, por lo que sería lógico pensar un criterio que se base en buscar distancias bajas entre los nodos. Con esta idea se desarrolla el siguiente método de búsqueda de una solución inicial:

1.- Para el nodo $i \in \{1, ..., n\}$, descartamos i en general (no consideramos enlaces) y, en caso de que i ya esté integrado en una ruta, todos los nodos que ya estén definidos en una ruta diferente a la del nodo i, además del nodo 1. Si éste ya tiene tantas llegadas como rutas queremos establecer, también queda descartado. Entre el resto de nodos, buscamos el arco de menor coste. En caso de empate, nos quedamos el arco que una i con el nodo de número más bajo.

2.- Una vez comprobados todos los nodos, nos fijamos en aquellos que dan el mismo arco no orientado. Esto significa que para $i, j \in \{1, ..., n/i \neq j\}$ el arco de menor distancia de i es el que se une con j y viceversa.

3.- Estos arcos seleccionados en el paso anterior se incluyen en nuestra solución. Esta inclusión puede darse de dos formas:

   a) El arco une al nodo 1 o a un arco que esté unido a este: Se tratará de un arco orientado para llegar a este nodo, y debemos reflejarlo. Indicamos, también, la ruta a la que pertenecerá este arco (en el grafo del siguiente ejemplo se observará de manera visual).

b) El arco une dos nodos sin ningún enlace a un arco orientado: Se representarán como arcos no orientados, a la espera de que, en futuras iteraciones, acaben enlazándose con un arco orientado, el cual les dará orientación.

4.- Tomamos una lista de aquellos nodos que no pertenecen a dos arcos distintos en la solución parcial provisional. El nodo 1 lo incluiremos si el número de arcos a los que se enlaza es menor al número de rutas. A partir de esta lista, repetimos los pasos anteriores para los elementos de la lista.

Una vez establecido este criterio, lo que nos queda es aplicarlo. Esto lo haremos con la ayuda del software R. Para ello, tendremos que aplicar algunas modificaciones a lo explicado anteriormente. La principal es que tendremos que orientar temporalmente los arcos del paso 3b. Para ello, estableceremos el principio que vamos del nodo menor al mayor y cambiaremos su orientación en caso de ser necesario. A partir de aquí, establecemos una segunda lista, la de los nodos que no tienen llegada tanto definitiva como temporal. Así, la elección de arcos siguiente será entre los nodos no enlazados con dos arcos y aquellos que no tienen un arco de llegada. El código utilizado para esta solución inicial aparece en el Apéndice C.

En cuanto a los nodos con dos arcos, tendremos que decidir como considerar su orientación provisional. La solución en R se irá construyendo como una matriz que indica por filas el nodo de salida del arco, el de llegada, su coste y la ruta a la que se le asigna (0 en caso de no tener ruta asignada). Por lo tanto, si en la misma columna tenemos nodos distintos del nodo 1 repetidos, significa que estamos considerando ese nodo como salida o llegada dos veces, hecho que no tiene sentido.

5.1.1. Ejemplo

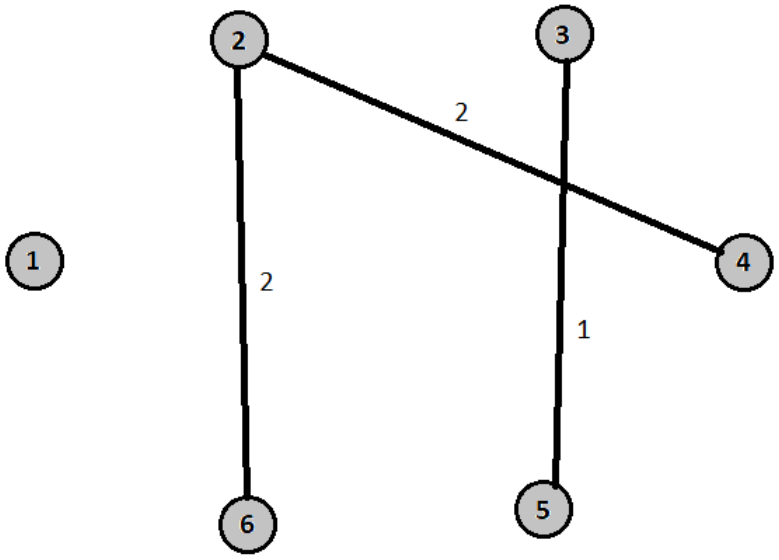
El ejemplo aquí presentado se ha realizado paso a paso para mostrar mejor lo que se va realizado en cada iteración. En general, esta búsqueda está automatizada con un algoritmo que tiene como criterio de para la ausencia de filas de ceros en un matriz que se reserva con tantas filas como arcos nos indica la restricción 1 que van a completar el árbol deseado, es decir $n-1$ arcos.

Volviendo a las distancias antes presentadas, lo primero que debemos hacer es comprobar las menores distancias en cada fila que no estén en la diagonal de la matriz. Los arcos con costes resultantes en esta primera iteración son:

Arcos de menor coste	Coste de los arcos	Arcos de menor coste	Coste de los arcos
(1,30) (1,42)	3	(23,35) (23,42)	2
(30,37)	1	(35,23)	2
(37,30)	1	(42,23) (42,30) (42,37)	2

De aquí nos quedaremos con los arcos (23,35), (23,42) y (30,37), con coste 2 en los dos primeros arcos y coste 1 en el último. El grafo en este caso quedará como sigue:

Búsqueda de solución inicial. Lunes ruta 1. Primera iteración.



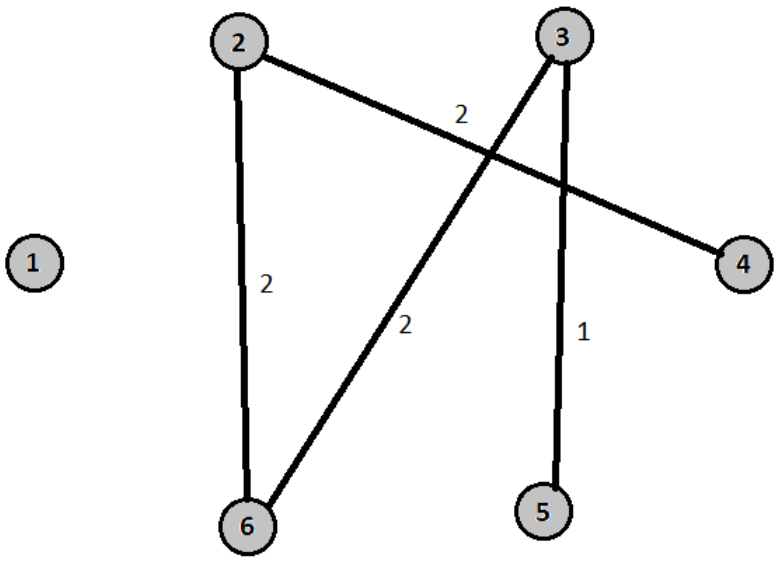
1=CHUS (1); 2=Esteiro (23); 3=Lousame (30);
4=Muros (35); 5= Noia (37); 6=Outes (42).

De estos nodos podemos descartar para la siguiente iteración el nodo de Esteiro. Además, para la aplicación en R consideramos que los nodos de llegada son el nodo de Muros y el de Noia. Con esto pasamos a la segunda iteración, la cual nos da los siguientes resultados:

Arcos de menor coste	Coste de los arcos	Arcos de menor coste	Coste de los arcos
(1,30) (1,42)	3	(30,42)	2
(35,42)	3,5	(37,30)	1
(42,30)	2		

Con estas condiciones, el arco a incluir es el (30,42), de coste 2. El grafo queda como sigue:

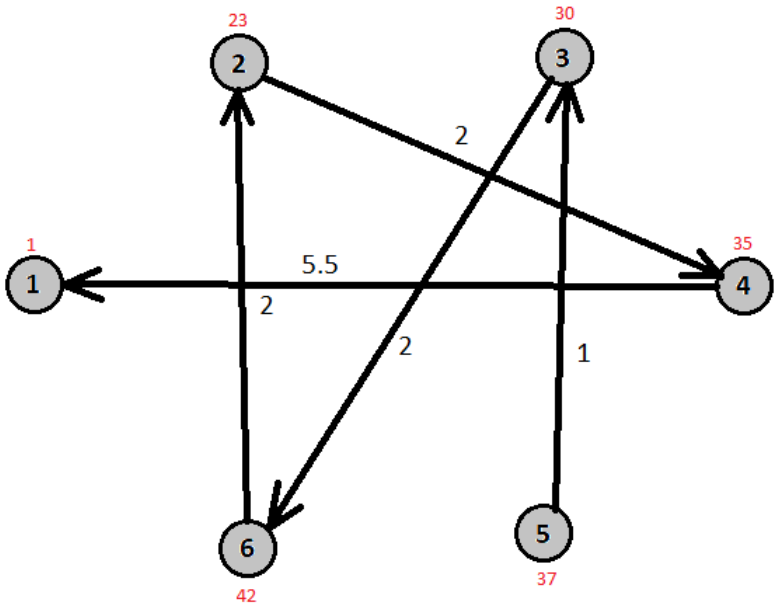
Búsqueda de solución inicial. Lunes ruta 1. Segunda iteración.



1=CHUS (1); 2=Esteiro (23); 3=Lousame (30);
4=Muros (35); 5= Noia (37); 6=Outes (42).

Por lo tanto, los nodos que no necesitamos tener en cuenta en la próxima iteración serán los de Esteiro, Lousame y Outes. Además, solo nos queda por enlazar el nodo CHUS, pues un enlace entre Muros y Noia provocaría un ciclo sin llegar al laboratorio para el análisis de las muestras. Esto, combinado a que tomamos el nodo de Noia como nodo de llegada provisional provoca que el posible enlace que nos quede sea el (35,1) entre Muros y el CHUS, con un coste de 5,5 unidades de tiempo. Además, este arco está orientado por la restricción que obliga al nodo CHUS a ser un nodo de llegada. Con esto, al unirse al resto de arcos ya establecidos, provoca la orientación definitiva del resto de arcos, quedando de la siguiente manera.

Solución inicial propuesta por nuestro método. Lunes ruta 1.



1=CHUS (1); 2=Esteiro (23); 3=Lousame (30);
4=Muros (35); 5= Noia (37); 6=Outes (42).

El resultado de esta solución es de 12,5 unidades de tiempo, por lo que, traducido en tiempo, será una ruta 45 minutos más rápida que la solución propuesta sin estudiar los tiempos entre nodos. Por otro lado, al utilizar AMPL vemos que el objetivo de nuestro problema aplicado a este conjunto de nodos es de 10,5 unidades de tiempo, lo cual, si bien es menor, el resultado obtenido con nuestro algoritmo de solución inicial no sería tan malo en caso de no conseguir mejorarlo.

Con respecto a este criterio, podemos comentar varias cosas. El primero es el comportamiento en caso de empates. Por defecto, en caso de empate, nos quedamos con los nodos de número más bajo. Esto no tiene por qué ser lo más adecuado, aunque es probable que lo consigamos arreglar a la hora de hacer las modificaciones posteriores. Por otro lado, las orientaciones provisionales nos limitan a la hora de poder conseguir nuevos arcos. Sin embargo, nos ayudan mucho a la hora de evitar ciclos, por lo que sigue siendo preferible mantenerlo.

Por último, cabe destacar que, en este ejemplo, el enlace del nodo CHUS resulta ser el último. Esto no tiene por qué suceder en otros casos, lo que nos dará una orientación clara para varios arcos y eliminará posibles problemas a la hora de buscar esta solución. En este caso, por ejemplo, como la orientación provisional para el nodo de Noia lo ponía como un nodo de llegada, el único arco posible que quedaba para el nodo CHUS era el de Muros, el cual es 1,5 unidades de tiempo más lento que el de Noia.

Con todo esto, queda claro que podríamos tener distintos criterios a la hora de buscar una solución inicial y pueden ser de ayuda en caso de no conseguir cambios en las rutas y comprobar si podemos mejorar más nuestras soluciones. También hay que tener en cuenta que este paso es opcional y podríamos partir de las rutas que ya tenemos como solución inicial y ver qué cambios

pueden llevarse a cabo. Además, podríamos tener también una elección inicial mixta, a partir de la solución ya conocida y corregir añadiendo o quitando los nodos que buscaríamos reubicar.

5.2. Cambios de arcos en la misma ruta

Una vez tenemos una solución inicial, tendremos que hacer modificaciones en ésta para mejorar el objetivo de nuestro problema mientras sigue cumpliendo las restricciones. El primer conjunto de modificaciones son aquellos que modifican la ruta. Este tipo de cambios son los más básicos y serán los que más se utilicen. Consideraremos tres cambios básicos:

- Cambiar el final de la ruta: Significa comprobar si hay arcos de menor valor que enlacen al nodo CHUS, y comprobar si, tras los cambios necesarios para cumplir con las restricciones, tenemos un objetivo menor al que hay con la solución vigente. En principio, tomaremos la que menos distancia tenga antes de pasar al siguiente cambio. Esos cambios consisten en la eliminación del anterior arco que acaba la ruta, la eliminación del arco de llegada al nuevo enlace al CHUS, que se sustituye por un arco entre su nodo de procedencia y el anterior nodo de llegada al CHUS y los cambios de orientación correspondientes para dar un árbol correcto.

- Cambiar un arco intermedio: Se refiere a, entre los nodos distintos al CHUS, ver si hay un arco de menos coste al que ya tenemos. El cambio lo haremos con el arco de menor coste y comenzaremos a hacer estos cambios desde el nodo más bajo hasta el más alto (empezamos con el 2, después el 3, etc.) que tengan esta posibilidad de cambio. Haciendo los cambios necesarios de arcos y orientación podemos tener hasta cuatro tipos de cambios distintos, si bien algunos podrían ser equivalentes. Hay que tener en cuenta también la obligación de llegada al nodo CHUS, que nos va a ayudar a orientar el resto y a comprobar si un arco nuevo debe orientarse de una forma o de la otra.

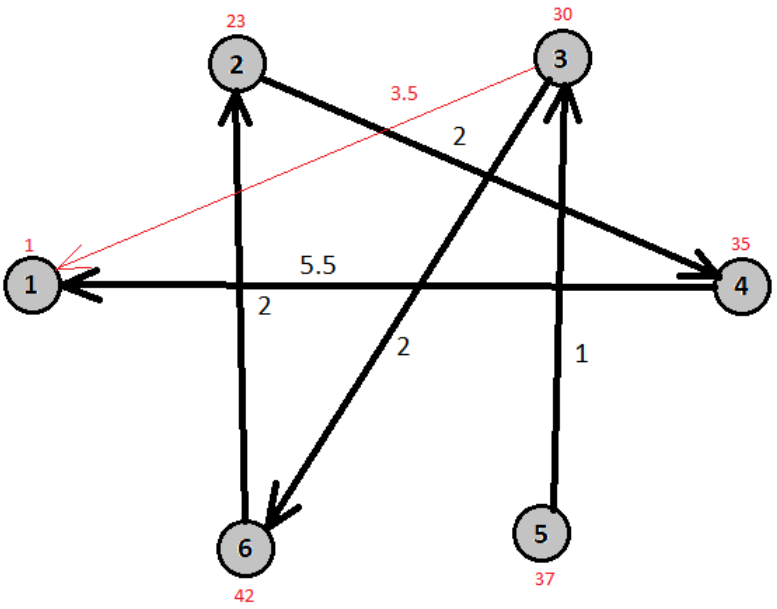
- Cambiar un arco de inicio: Significa cambiar de orientación el arco que inicia una ruta. Para ello, lo que tendremos que hacer es enlazar el que sería el tercer nodo de la ruta con el nodo que iniciaba la ruta antes de cambiar la orientación a este primer arco de la ruta, con lo que se elimina el arco que unía los nodos que ocupaban el segundo y tercer lugar de la ruta antes de la modificación. En este caso, un objetivo igual o incluso un poco superior se puede admitir para repetir los cambios anteriores y encontrar una solución mejor que la que teníamos en un principio.

5.2.1. Ejemplo de cambios en la misma ruta: Arco con nodo 1

A diferencia de como hemos trabajado al construir la solución, los distintos cambios que vayamos proponiendo no están unidos en un algoritmo único, si no que se realizan, de momento, de una manera más manual. Sin embargo, no se descarta que se pueda escribir un código que consiga realizar todos estos cambios en un mismo algoritmo, estableciendo criterios de paradas, distinguiendo los casos en los que nos encontremos, etc.

Comenzando con el cambio en el nodo de llegada, analizando las distancias de tiempo al nodo CHUS, vemos que hay dos nodos que tienen la menor distancia a éste: El de Lousame (30) y el de Outes (42). En este ejemplo nos centraremos en el primero para aclarar como aplicar los cambios.

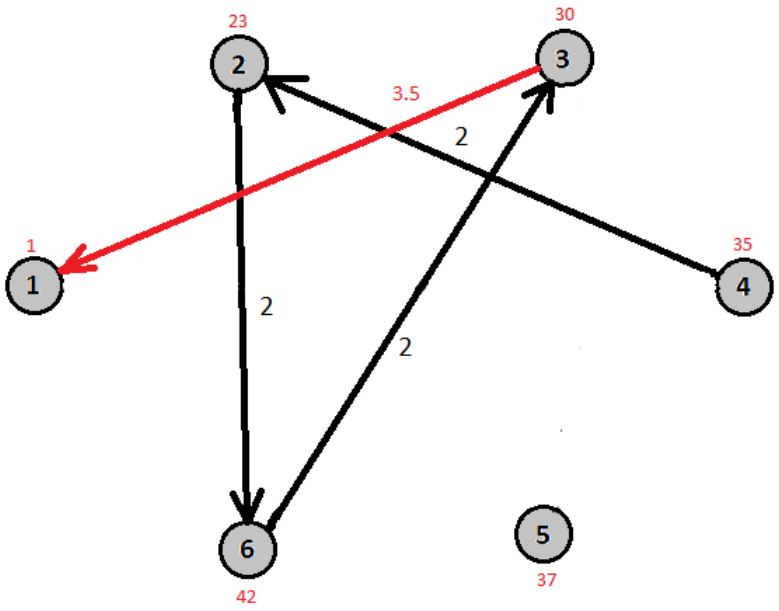
Ruta dada por la solución inicial con el nuevo arco propuesto de llegada al CHUS.



1=CHUS (1); 2=Esteiro (23); 3=Lousame (30);
4=Muros (35); 5= Noia (37); 6=Outes (42).
En rojo, el arco que queremos introducir. Se tratará de un arco orientado al contener el nodo 1.

Tenemos un ejemplo de por qué hacemos los cambios como hemos explicado. Si quitamos el arco de salida desde Lousame, lo cual aparenta ser lo lógico, nos queda un ciclo entre Esteiro, Muros y Outes si en dicho arco cambiamos el origen por el anterior predecesor del CHUS. Por lo tanto, lo que nos quedará es hacer una reorientación para que suceda un cambio simple. Eliminando los arcos sobrantes del grafo, obtenemos la siguiente ruta:

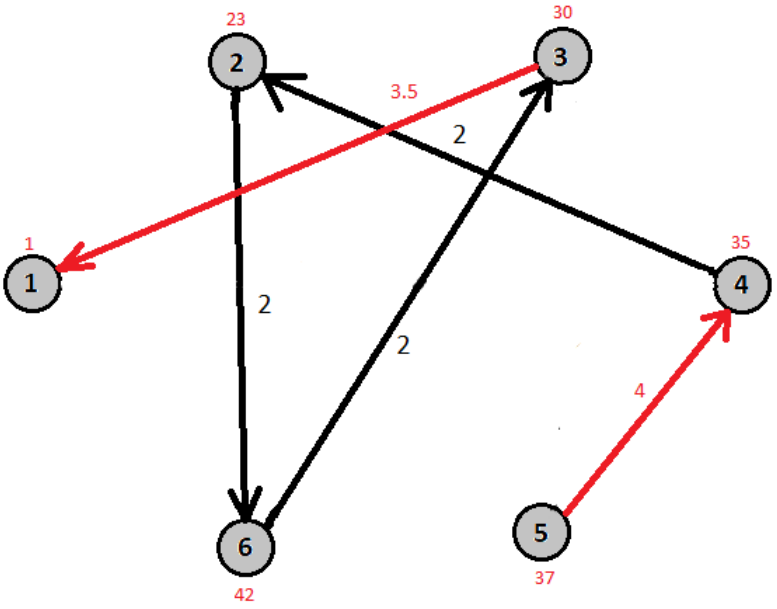
Ruta provisional después de añadir el arco propuesto de llegada al CHUS y eliminar los arcos sobrantes.



1=CHUS (1); 2=Esteiro (23); 3=Lousame (30);
4=Muros (35); 5= Noia (37); 6=Outes (42).
En rojo, el arco que hemos acabado de introducir.

Como podemos observar, en este caso todos los nodos están enlazados menos el nodo de Noia. Además, salvo el nodo de Muros, el resto de nodos están conectados a la cantidad de arcos máxima para nuestro problema. Por lo tanto, cabe esperar que el arco orientado que falta por añadirse al grafo sea el que va de Noia a Muros, quedando la ruta como se muestra a continuación:

Ruta resultante de añadir el arco propuesto de llegada al CHUS y el seleccionado para completar la ruta.



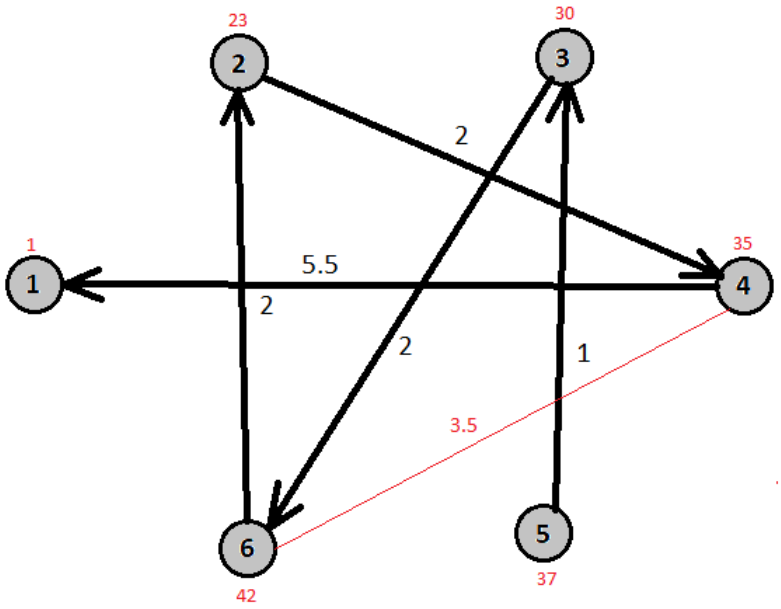
1=CHUS (1); 2=Esteiro (23); 3=Lousame (30);
4=Muros (35); 5= Noia (37); 6=Outes (42).
En rojo, los arcos que hemos acabado de introducir.

Sin embargo, esta ruta no logra mejorar el objetivo obtenido en la solución inicial, pasando a un coste de 13,5 unidades de tiempo, 10 minutos más que en el caso anterior. Así, pues, en este caso volveríamos a la solución inicial y comenzaríamos con el siguiente cambio. Conste añadir que, de forma un poco más rebuscada, podríamos cambiar el arco de salida del nodo de Lousame y evitar el ciclo tomando el camino entre Noia y Outes, con la orientación que corresponda. Por consiguiente, podría ser posible, aunque no tan simple, hacer el cambio de esta forma.

5.2.2. Ejemplo de cambios en la misma ruta: Arcos intermedios

Lo siguiente a realizar en este caso será ver los nodos intermedios y comprobar si pueden tener un arco de menor coste y trabajar a partir de éste los casos correspondientes para observar si hay una mejora en el objetivo. En nuestro caso, sólo tenemos indicada mejora con el nodo de Muros, donde se recomienda un enlace con Outes. En caso de ver mejoras en otros nodos, haremos como se ha explicado, comenzando con el nodo de etiqueta más baja y siguiendo hasta que, al pasar por todos los disponibles, no podamos obtener mejoras.

Ruta dada por la solución inicial con el nuevo arco propuesto para incluir.

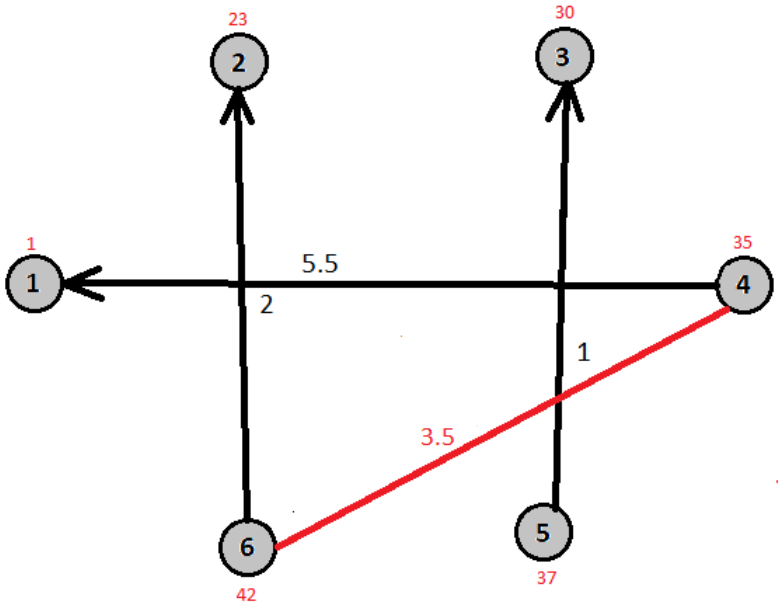


1=CHUS (1); 2=Esteiro (23); 3=Lousame (30);
4=Muros (35); 5= Noia (37); 6=Outes (42).
En rojo, el arco que queremos introducir.

Como podemos observar, el hecho de incluir este arco en el árbol provoca que tengamos que quitar arcos en los nodos que se unen, así como incluir un nuevo arco que complete el mismo. En este caso, incluir este arco no da ninguna mejora en el objetivo, por lo que tendremos que pasar al siguiente cambio. Sin embargo, ejemplificaremos su funcionamiento con uno de los posibles cambios.

En este caso, quitaremos el arco que unen Lousame y Outes y el que une Esteiro con Muros, dejando el árbol provisional de la siguiente forma:

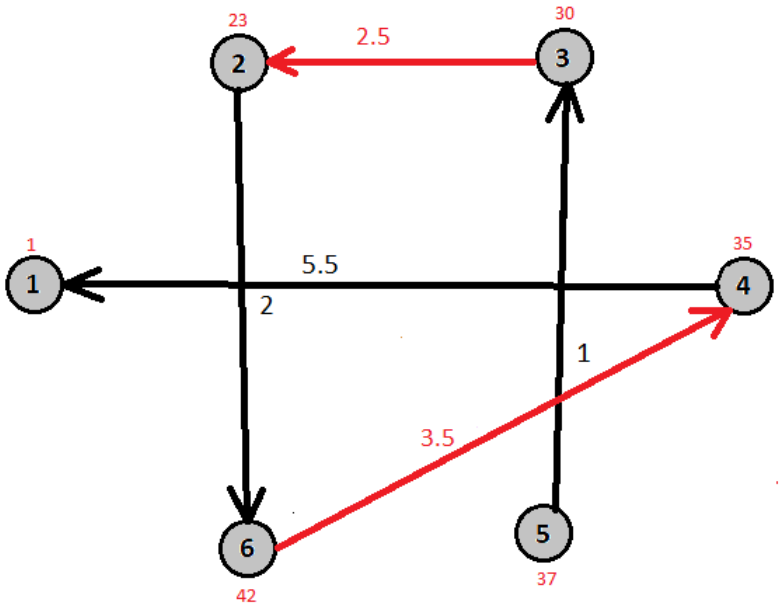
Ruta provisional después de añadir el arco propuesto y eliminar los arcos correspondientes.



1=CHUS (1); 2=Esteiro (23); 3=Lousame (30);
4=Muros (35); 5= Noia (37); 6=Outes (42).
En rojo, el arco que queremos introducir.

En este caso tenemos dos posibles enlaces, uno que une el nodo de inicio de ruta, el cual cambiaría el nodo de inicio y el que une los nodos de los cuales hemos quitado los arcos, como ya hemos hecho previamente para incluir el arco que hemos buscado como candidato a mejorar el objetivo. El cambio básico se produciría manteniendo el inicio de la ruta en Noia, como en la solución de la que partimos. Así, el arco con el que vamos a completar este árbol será el que enlaza Lousame y Esteiro. Con los arcos orientados, el nuevo árbol queda de la siguiente forma:

Ruta resultante de añadir el arco propuesto y el seleccionado para completar la ruta.



1=CHUS (1); 2=Esteiro (23); 3=Lousame (30);
4=Muros (35); 5= Noia (37); 6=Outes (42).
En rojo, los arcos que hemos introducido con estos cambios.

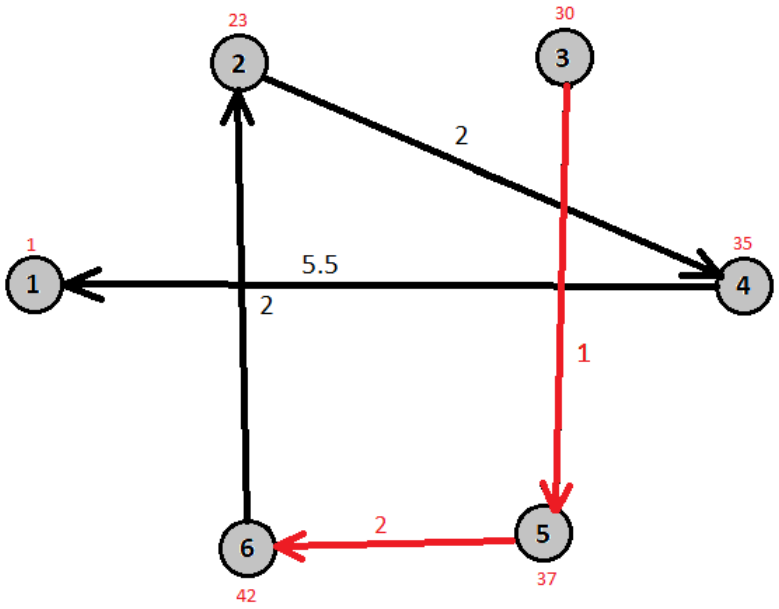
Con esta solución, el objetivo pasa a ser de 14,5 unidades de tiempo, que es 20 minutos más lenta que la inicial. En caso de haber cambiado el inicio de la ruta, como mucho obtendríamos una solución de mismo objetivo. Así, tendríamos que ver otra clase de cambios posibles e incluso ver la posibilidad de cambiar el nodo de inicio de ruta. Como añadido, decir que, cuando estos cambios afecten al nodo CHUS solamente tendremos en cuenta este tipo de cambios cuando no afecte a la orientación del mismo. De la manera como trabajamos en R, un cambio de orientación con este nodo implica un cambio de posición, por lo que devolverlo a la posición anterior resultaría tedioso y mantenerlo en la posición que ocupe en la lista va a resultar en problemas a la hora de hacer los cambios de orientación en el software.

Por lo tanto, vamos a hacer el último cambio que hemos explicado: el cambio de inicio de ruta.

5.2.3. Ejemplo de cambios en la misma ruta: Cambio en el inicio de ruta

En este caso, en vez de iniciar la ruta en Noia la iniciaremos en Lousame y veremos qué pasa, en el sentido de si pueden aparecer nuevas opciones de cambio de arcos al aplicar los anteriores. Así, pues, la ruta queda entonces de la siguiente forma:

Ruta resultante de cambiar la orientación del arco que inicia la solución de partida.



1=CHUS (1); 2=Esteiro (23); 3=Lousame (30);
4=Muros (35); 5= Noia (37); 6=Outes (42).
En rojo, los arcos que hemos cambiado.

Con este cambio el valor del objetivo es el mismo, 12,5 unidades de tiempo. Sin embargo, al aplicar sobre esta ruta el cambio resultante de borrar el arco que une el CHUS con Muros y estableciendo Lousame como la localidad que procede a la llegada al CHUS, con las modificaciones de orientaciones correspondientes, logramos una ruta que resulta óptima, dando un coste de 10,5, el mismo que se obtuvo cuando se llevó este problema a AMPL

Para concluir este apartado, cabe destacar que podríamos incluir la opción de elegir cual de los dos nodos que quedan libres seleccionamos para el enlace. Recordar que, al realizar cualquiera de los dos primeros cambios, al eliminar un arco y tener elegido el nodo de partida nos quedarán uno o dos nodos sin tener dos arcos enlazados y con los que podemos reenlazar al resto de la ruta el conjunto de nodos entre estos extremos que han quedado aislados durante el cambio, y pudiendo llevar a soluciones mejores de manera más rápida o a obtener soluciones que no nos permitan una mejora posterior aunque sí obtengamos una mejora local del tiempo de coste.

5.3. Cambios de arcos en rutas distintas

Una vez vistas las posibilidades implementadas para los cambios dentro de una misma ruta, pasamos a observar qué podremos hacer entre rutas. Como antes, buscaremos una forma de realizar cambios de la manera más básica posible, aunque podría resultar un poco más complejo que las sustituciones realizadas anteriormente. Los cambios de este apartado los podemos agrupar en dos tipos: Los cambios intermedios y los cambios de inicio de ruta.

Empezando con los cambios intermedios, este tipo de cambios serán muy similares a los cambios de arcos intermedios para nodos de la misma ruta: Entre los nodos distintos al nodo CHUS, vamos comprobando si existe un arco de menor coste para ese nodo. Esto puede generar problemas, pues es el mismo proceso para ambos casos. Así, pues, lo que podemos hacer para solventar esta confusión es separar en casos con la misma ruta y con rutas distintas. La conexión final del nuevo árbol consistirá en, por una parte, unir los nodos que estaban unidos al nodo que cambia de ruta y, en segundo lugar, este tendrá que acabar en la nueva ruta de manera que lo conectemos al nodo deseado y a cualquiera de los dos nodos que unen al otro nodo del nuevo arco. Para ello, la unión se hará con aquel nodo que de un arco de menor coste, como resulta lógico pensar pues está en la línea de todos los razonamientos utilizados en nuestro proceso.

Por otro lado, podemos realizar este mismo cambio pero solo hacia los extremos iniciales de cada ruta. El proceso sería similar al anterior: Se comprueba el nodo de menor distancia a cada extremo, se lleva este nodo como nuevo extremo de la ruta y los nodos unidos a éste se unen en un nuevo arco. Lógicamente, si resulta que ese nuevo nodo es el extremo de otra ruta, este proceso de unión no es necesario, al no quedar nodos no enlazados con el CHUS.

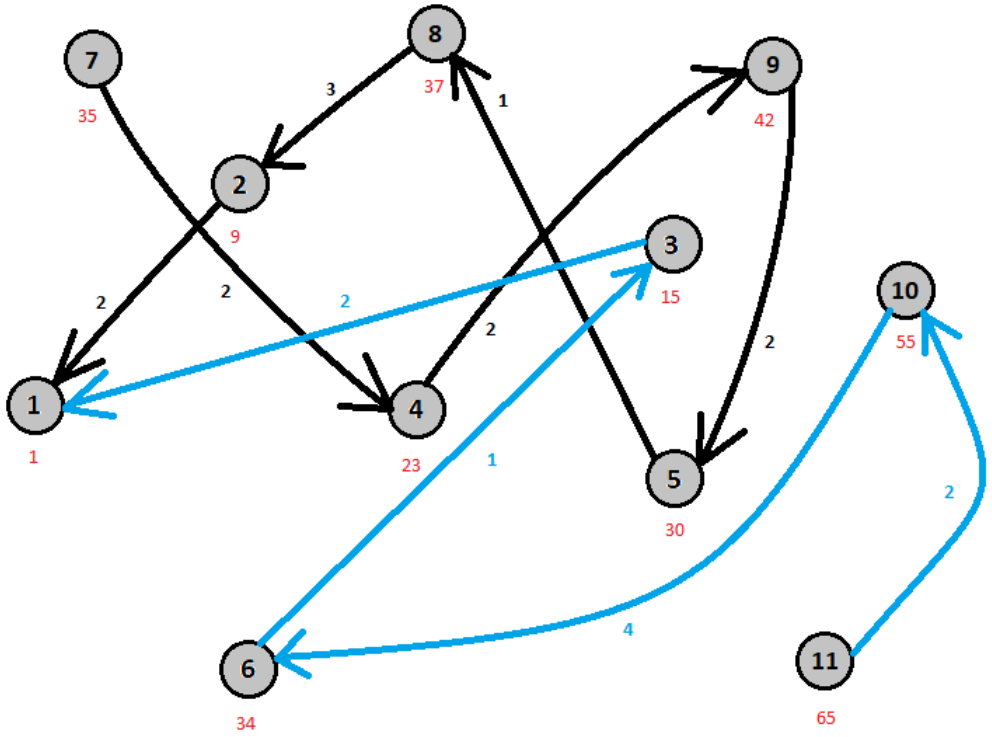
Es necesario tener en cuenta que tenemos la restricción 11 que limita el tiempo de las rutas, y la cual sería recomendable cumplir. Así, pues, sería comprensible que, en estos cambios, diéramos prioridad a las rutas de menor tiempo, pues las rutas con mayor coste posiblemente se pasen de este límite, pues en este punto del método aún no lo hemos tenido en cuenta pero se tendrá en consideración en el siguiente apartado. Por lo tanto, podemos ver esto como una manera de ahorrarnos problemas y posibles cambios posteriores si ya llevamos los pasos realizados de esta manera.

5.3.1. Ejemplos de cambios en rutas distintas

Pasamos, ahora, a ver un ejemplo del funcionamiento de estos cambios. Como es lógico, necesitaremos de, al menos, dos rutas para poder ejemplificar estas modificaciones. Para esta ocasión, vamos a trabajar con los nodos que conforman actualmente las rutas 1 y 3 del lunes. Las localidades con las que trabajaremos en esta ocasión serán Bertamiráns (9), Calo (15), Esteiro (23), Lousame (30), Milladoiro (34), Muros (35), Noia (37), Outes (42), Santa Comba (55) y Valdubra (65), además del nodo CHUS (1). El funcionamiento de estos cambios los veremos a partir de la solución inicial dada por el método que hemos propuesto al inicio del capítulo, aunque podría ser recomendable hacer los cambios dentro de la misma ruta antes de comenzar a cambiar nodos de ruta.

Con las condiciones que hemos explicado, la solución inicial que tenemos nos dará las dos rutas que aparecen en el siguiente grafo:

Solución inicial propuesta por nuestro método. Lunes rutas 1 y 3.

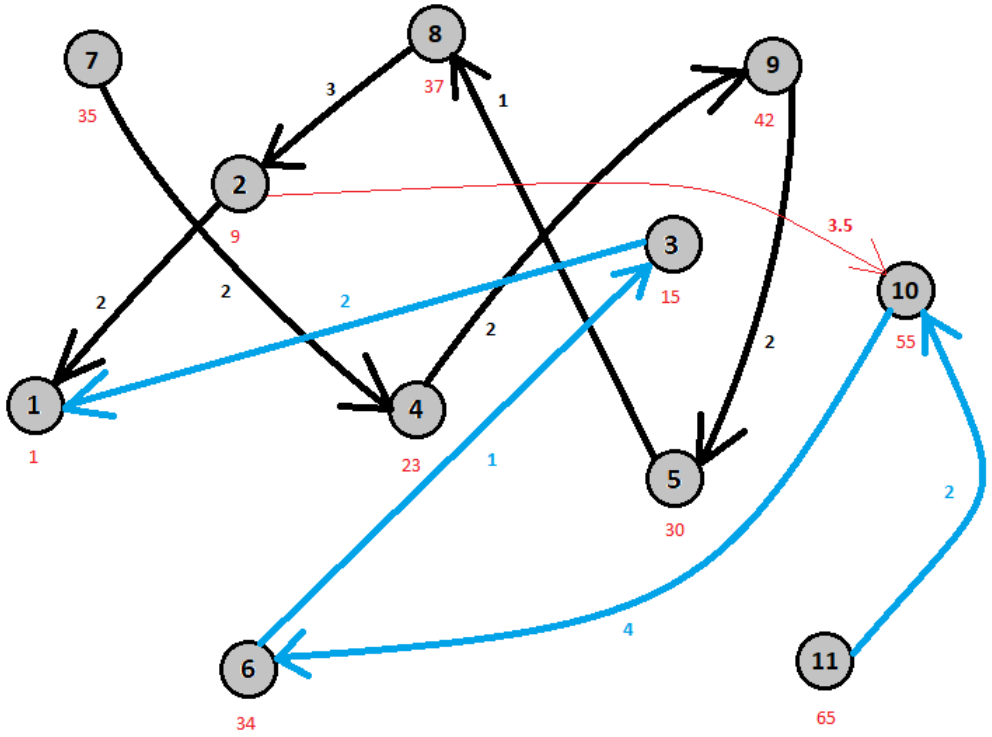


1=CHUS (1); 2=Bertamiráns (9); 3=Calo (15); 4=Esteiro (23);
5=Lousame (30); 6=Milladoiro (34); 7=Muros (35); 8=Noia (37);
9=Outes (42); 10=Santa Comba (55); 11=Valdubra (65)
En negro, la ruta 1. En azul, la ruta 2.

Con esta solución tenemos un coste total de 21 unidades de tiempo, repartidas en 12 periodos de tiempo para la ruta 1 y los 9 periodos de tiempo restantes como coste de la ruta 2.

Comenzamos con el cambio intermedio. En este caso, al ver el coste de los distintos arcos posibles, tenemos que el arco que une las localidades de Bertamiráns y Santa Comba es de menor coste que los arcos actuales para la localidad de Santa Comba. Al establecer un límite de tiempo para las rutas, el criterio que utilizaremos para estos cambios será el de llevar el nodo en la ruta de mayor coste a la de menor. En este caso, en consecuencia, intentaremos llevar Bertamiráns a la ruta 2, como aparece en el siguiente grafo:

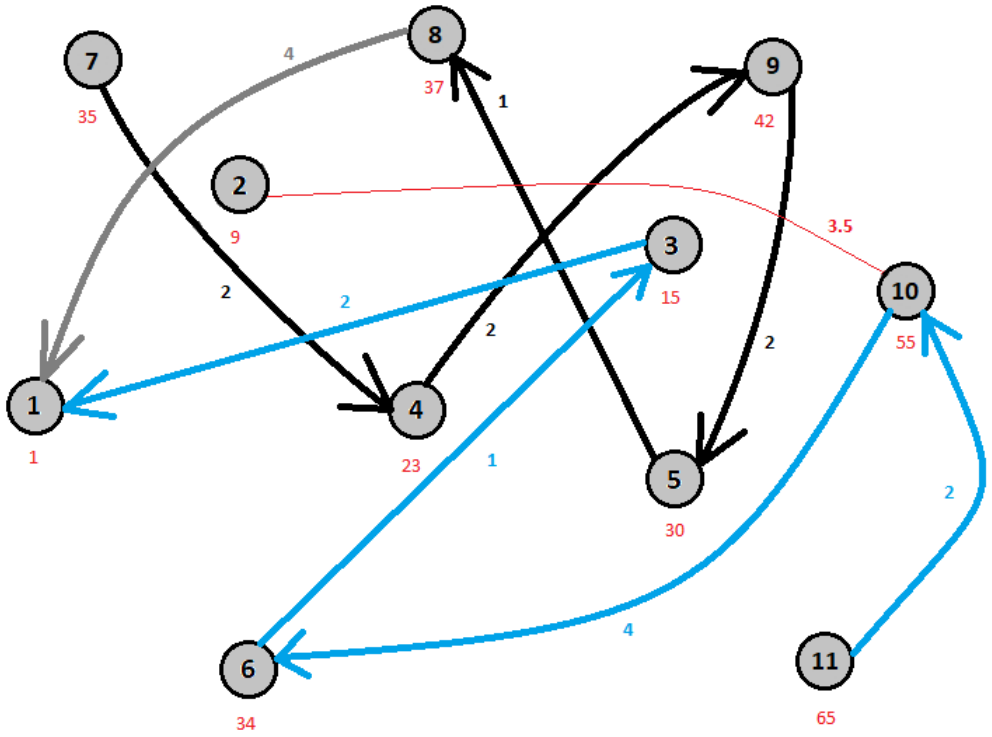
Presentación del cambio de ruta para Bertamiráns.



1=CHUS (1); 2=Bertamiráns (9); 3=Calo (15); 4=Esteiro (23);
5=Lousame (30); 6=Milladoiro (34); 7=Muros (35); 8=Noia (37);
9=Outes (42); 10=Santa Comba (55); 11=Valdubra (65)
En negro, la ruta 1. En azul, la ruta 2.
En rojo, el arco que queremos incluir. La flecha en este caso señala hacia la ruta que irá el nodo cambiado.

Como queremos quitar nuestro nodo 2 de la ruta 1, vamos a tomar el arco que une los dos nodos que conectan este nodo. De esta manera, esta ruta seguirá cumpliendo nuestras restricciones. Nos quedará un grafo con el nodo 2 aislado del resto a falta de incluirlo, esta vez, en la ruta 2. En caso de que este nodo sea inicio de ruta, este paso no será necesario y solo eliminaremos el arco que une este nodo a la ruta.

Ruta 1 resultante de pasar Bertamiráns a la otra ruta.

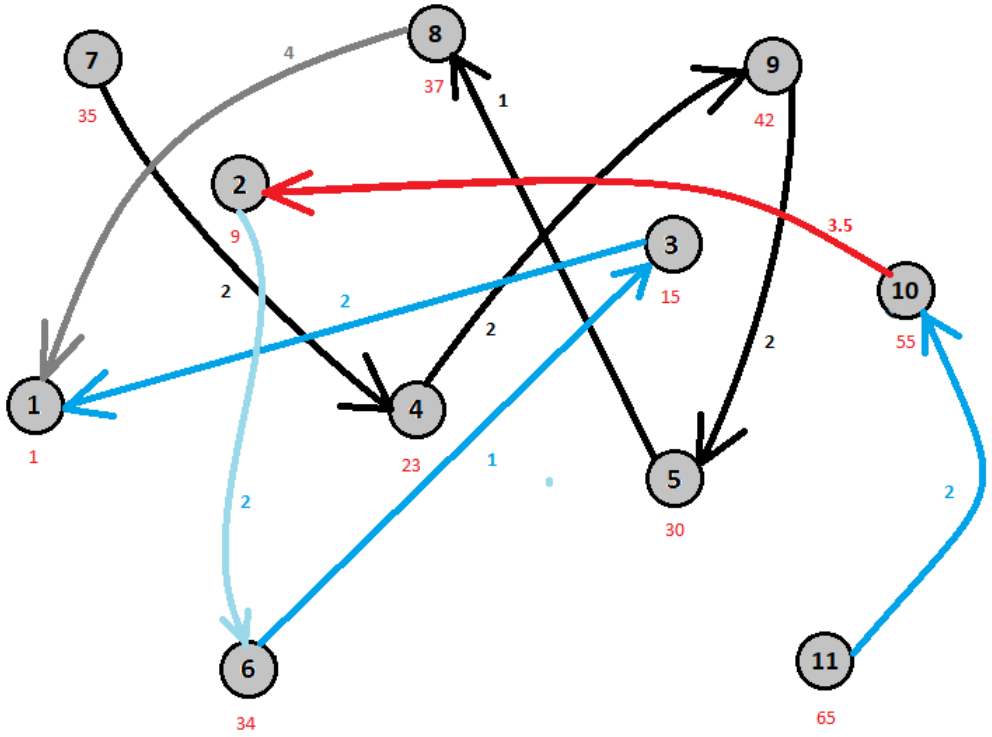


1=CHUS (1); 2=Bertamiráns (9); 3=Calo (15); 4=Esteiro (23);
5=Lousame (30); 6=Milladoiro (34); 7=Muros (35); 8=Noia (37);
9=Outes (42); 10=Santa Comba (55); 11=Valdubra (65)
En negro, la ruta 1. En azul, la ruta 2.
En rojo, el arco que queremos incluir.
En gris, el nuevo arco formado a causa del arco que deseamos incluir.

El siguiente paso va a depender de si el nodo al que queremos unir nuestro nodo aislado es de inicio de ruta o no. En caso de ser de inicio de ruta, en este cambio no consideraremos que el nodo nuevo entre como nuevo inicio de ruta, pues este caso quedará para el siguiente cambio a explicar. Por otro lado, que es el que sucede en el caso con el que estamos ejemplificando este tipo de cambios, el nodo está en medio de la ruta. Entonces podemos tener dos situaciones: que el nodo aislado ocasione menos coste siendo el nodo de salida al nodo de la ruta o que el coste sea más bajo cuando es el nodo de llegada al nodo que tenemos en la ruta.

En este caso, el nodo 10 está enlazado al nodo 11 (procedencia) y al nodo 6 (llegada). Tendremos que ver los costes de los arcos que enlazan cada uno de estos nodos con el nodo 2. El coste de ir de 11 a 2 es de 3,5 unidades de tiempo, mientras que el arco de 2 a 6 tiene un coste de 2. Por lo tanto, el otro arco de enlace en esta ruta para el nodo 2 deberá unir este nodo con el nodo 6, con lo que suprimimos el arco que une el nodo 10 con el 6 para incluir el nodo 2 entre ambos. De esta manera, las rutas quedarán de la siguiente forma:

Solución resultado de cambiar de ruta a Bertamiráns.



1=CHUS (1); 2=Bertamiráns (9); 3=Calo (15); 4=Esteiro (23);
5=Lousame (30); 6=Milladoiro (34); 7=Muros (35); 8=Noia (37);
9=Outes (42); 10=Santa Comba (55); 11=Valdubra (65)
En negro, la ruta 1. En azul, la ruta 2.
En rojo, el arco que queremos incluir.
En gris (1) y azul claro (2), los nuevos arcos formados a causa del arco que deseamos incluir.

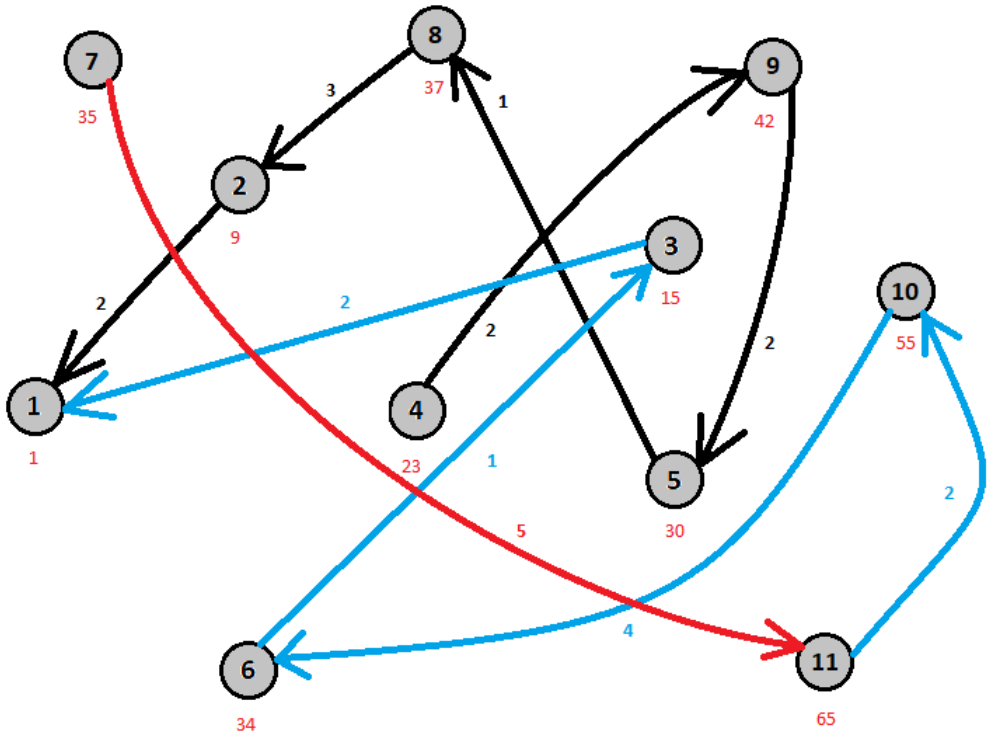
La solución actual sería de un coste de 21,5 unidades de tiempo, lo cual sería 0,5 unidades de tiempo más que la solución de la que procedemos. Sin embargo, hay varias situaciones por las cuales podríamos preferir trabajar a partir de esta solución y no volver a la anterior. La primera situación sería condicional al hecho de no haber mejorado la solución con otros cambios y pensando que sí podríamos mejorar el coste final. En ese caso, sería recomendable trabajar a partir de esta solución y ver si podemos lograr mejores soluciones para el problema que estamos tratando, lo que se traduce en salir de un óptimo local en busca de otro mejor o, incluso, un óptimo global.

Otra situación en la que podemos preferir esta solución es en la de mejorar el tiempo de una ruta sin perjudicar en exceso el tiempo de otra. Con la solución inicial tenemos que la ruta 1 tardará 12 en realizarse, mientras que la 2 sólo requiere de 9. Con este cambio, ahora la ruta 1 tarda en realizarse 11 unidades de tiempo, costando 10,5 para la ruta 2. Si no tuviésemos un límite de tiempo deseado para cada ruta este detalle sería menor. Sin embargo, con este límite podría ser recomendable quedarnos con la solución que nos dé rutas con tiempos bajos, y muy recomendable si estos tiempos están por debajo del límite establecido.

Pasamos ahora a los cambios en los comienzos de ruta. Tendremos dos casos con los que trabajaremos de manera análoga: Cuando el nuevo inicio de ruta es un nodo intermedio y cuando el inicio de ruta es el inicio de otra ruta. Los cambios que tendremos entre un caso y otro se basarán en completar las rutas. Además, estos cambios podremos considerarlos con dos tipos de elecciones: Utilizando el criterio con el que elegimos los cambios intermedios tanto en la misma ruta como en el cambio de ruta y eligiendo el arco de menor coste entre los inicios de ruta y los nodos de las otras rutas. El primer caso nunca nos va a dar arcos entre dos inicios de ruta y será utilizable luego de ver que, aplicado a un cambio intermedio (caso anterior), no tenemos mejora. Para comprobar la unión de dos nodos inicios de ruta tenemos la segunda opción, donde elegimos directamente entre los inicios de ruta.

Tomando cualquiera de estos criterios no obtenemos ninguna mejora en el ejemplo con el que estamos trabajando. En consecuencia, vamos a ejemplificar este cambio con los nodos que comienzan las rutas. La ruta 1 empieza en Muros, mientras que la ruta 2 partirá de Valdubra. Aplicando el criterio de eliminar el nodo de la ruta de mayor coste, nuestro objetivo será llevar la localidad de Muros a la ruta 2 y que inicie esta ruta. En este caso el cambio es realmente sencillo y sólo se hará un cambio de la siguiente forma:

Solución resultante de hacer que Muros pase a ser inicio de la ruta 2.



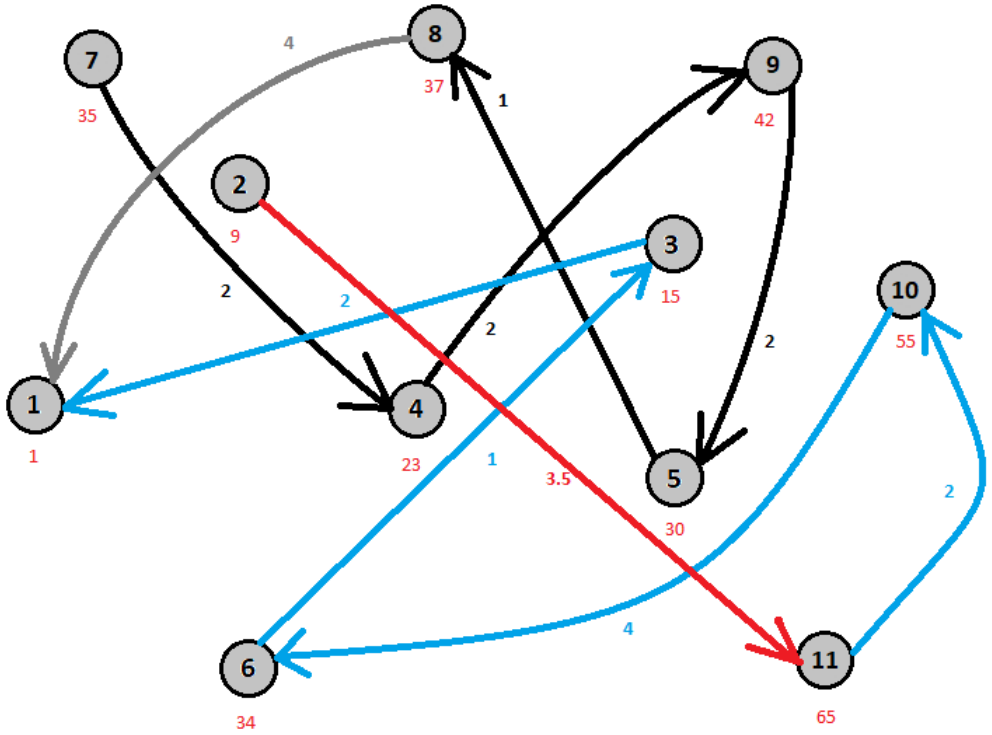
1=CHUS (1); 2=Bertamiráns (9); 3=Calo (15); 4=Esteiro (23);
5=Lousame (30); 6=Milladoiro (34); 7=Muros (35); 8=Noia (37);
9=Outes (42); 10=Santa Comba (55); 11=Valdubra (65)
En negro, la ruta 1. En azul, la ruta 2.

En rojo, el arco que vamos a incluir.

Este cambio ya podemos ver sencillamente que no va a resultar bueno. El coste total asciende a 24 unidades de tiempo, 3 más que la solución anterior. Además, los tiempos en las rutas también se ven perjudicados, pues si bien aliviamos un poco la ruta 1, quedando en 10 unidades de tiempo, la ruta 2 aumenta a 14, quedando por encima del límite que queremos y el cual deberemos tener en cuenta también para considerar válida la solución.

Veamos ahora el caso de que un nodo intermedio se haga inicio de ruta. Si nos volvemos a fijar en la localidad de Valdubra, entre las localidades de la ruta 1 el trayecto de menor coste lo unirá con Bertamiráns. Así, para cumplir este enlace, nos quedará unir los nodos que estaban enlazados aquí para poner Bertamiráns como localidad de inicio en la ruta 2, dejando las rutas como siguen:

Cambio de inicio de la ruta 2 a partir de los nodos de la ruta 1.



1=CHUS (1); 2=Bertamiráns (9); 3=Calo (15); 4=Esteiro (23);
5=Lousame (30); 6=Milladoiro (34); 7=Muros (35); 8=Noia (37);
9=Outes (42); 10=Santa Comba (55); 11=Valdubra (65)
En negro, la ruta 1. En azul, la ruta 2.
En rojo, el arco que vamos a incluir.

Con esta situación, el coste total es de 23,5 unidades de tiempo, por lo que podríamos descartar esta solución al ser peor que la inicial, como ya habíamos adelantado. Además, el coste de cada ruta nos confirma que la solución no es buena, pues la ruta 2 se extiende hasta las 12,5 unidades de

tiempo, pasando nuestro límite. Lo que solo podríamos saber trabajando a partir de esta solución es si nos puede llevar a una solución mejor que la de partida.

Como observación final, decir que, como en el caso dentro de la misma ruta, podríamos decidir si cambiar al nodo que inicia la ruta que cortamos o si quedarnos con el que tenemos, pudiendo llevar a soluciones mejores. De todas formas, un análisis más a fondo, observando si merece la pena hacer algo más complejo, podría ayudarnos al estudio de esta propuesta de cambio de rutas.

5.4. Cambios por límite de tiempo

Una vez vistos los cambios posibles en el propio grafo, nos queda revisar la restricción 12 de límite de tiempo de duración total de cada ruta. Los cambios aquí presentados buscan de manera natural intentar evitar que este límite sea superado. Sin embargo, esto no garantiza nada e incluso debemos considerar la posibilidad de que alguna ruta exceda este límite.

La opción que tendremos será sencilla: Revisar el coste de las rutas y ver aquellas que excedan el valor estipulado. Lo siguiente será partir desde el nodo CHUS e ir sumando hacia atrás en esa ruta hasta que la suma exceda el límite considerado. Este último nodo solo permanecerá en la ruta si la suma en ese punto llega al límite. En caso de pasarse, se quitará de la ruta, así como todos los nodos que lo precedan.

Una vez hecho esto, lo que nos quedará será recolocar los nodos en las otras rutas que no hayan alcanzado este límite, pudiendo permitir que se exceda un poco para, a partir de ahí, volver a aplicar los cambios de los apartados anteriores y ver si conseguimos una solución que cumpla con las restricciones y tenga un coste bajo.

Como hemos dicho antes, cabe la posibilidad de que las rutas excedan este límite. Esto se considerará a partir del criterio que elijamos: Un número de veces máximo que hagamos este recorte, que tengamos unas rutas con tiempos más o menos iguales, dejar que una ruta cargue con exceso pero el resto cumpla el límite... En caso de complicaciones, dependerá de nuestras exigencias y de nuestras necesidades el proceso a seguir para considerar como válida una solución.

5.5. Presentación de la solución que consideraremos

Llegados a este punto, tendremos que dar una presentación de la solución obtenida por el método heurístico, incluyendo en ésta los periodos de tiempo en los que alcanzamos cada nodo de salida. A la hora de trabajar en R para buscar una solución por el método propuesto, la presentación de esta solución se realiza en una matriz de cuatro columnas, donde, como se ha dicho antes, cada fila representa un arco, e indicando cada una de las columnas:

- **Columna 1:** Presenta los nodos de salida del arco, donde se efectúa la carga.
- **Columna 2:** Son los nodos hacia donde se dirige el vehículo una vez ha partido del nodo de salida.
- **Columna 3:** Aparecen los costes del arco, sumando el tiempo de carga del nodo de salida y el trayecto entre nodos.

- **Columna 4:** Ruta en la que se localiza el arco.

Comparado con la presentación de la solución explicada en el capítulo anterior, y que partía de la estructura de la solución dada por AMPL, vemos que tenemos una columna menos, la cual se corresponde con la columna de periodos de tiempo. Entonces, tendremos que preparar un poco la forma de presentar esta solución, como ya tuvimos que hacer con la solución a partir de AMPL, aunque de manera más simple. En este caso, duplicaremos la columna de los costes, quedando en las columnas 3 y 5. A continuación, sustituimos la columna 3 por ceros.

Esta columna 3 va a completarse con una presentación temporal de periodos de tiempo. Lo que haremos será poner los periodos de tiempo partiendo desde $t = 0$ en todos los casos. El nodo de salida de la ruta parte desde cero y se van sumando los tiempos de llegada a cada nodo, y aplicando la función suelo una vez que todas las sumas parciales se han realizado. Una vez efectuado esto y con los arcos ordenados de la forma adecuada, aplicamos la corrección ya realizada para las soluciones de AMPL, al tener la solución una estructura análoga a la que obtenemos al trabajar con AMPL. De esta manera, obtendremos los periodos de tiempo correspondientes a nuestras soluciones, y con las cuales podemos establecer las distintas horas de llegada estimadas para cada localización.

Por último, se puede pensar en sustituir los números de cada nodo por los números reales al trabajar con todas las localidades. Esto es debido a que para la búsqueda de la solución resultará más cómodo trabajar desde 1 hasta n que con un conjunto de números cualquiera.

Capítulo 6

Resultados

En este capítulo presentaremos una comparación de resultados (actuales, obtenidos con AMPL y obtenidos con R) de cada una de las rutas que se realizan a la fecha de redacción de este trabajo, separadas en tablas con los distintos días de la semana en los cuales las rutas son realizadas. Al final, se discutirán estos resultados.

En las tablas aparecerán, de izquierda a derecha, el número de ruta, la ruta actual, la ruta obtenida en AMPL y la obtenida en R con el método propuesto. Las rutas irán siempre acompañadas de los costes de cada una de las rutas presentadas. Las localidades reales a las que están asociados los nodos a los que se hace referencia en estas rutas aparecerán en el Apéndice A al final del trabajo.

6.1. Rutas individuales. Comparación con AMPL y actual

Rutas del lunes						
Número	Ruta actual	Coste	Ruta AMPL	Coste	Ruta R	Coste
1	35-23-42-37-30-1	10.5	Misma que la actual	10.5	Misma que la actual	10.5
2	48-49-70-8-51-52-45-11-50-43-1	19.5	Misma que la actual	19.5	48-49-70-8-52-51-45-11-50-43-1	19.5
3	65-55-9-34-15-1	10.5	55-65-9-15-34-1	10	Misma que en AMPL	10
4*	32-56-5-10-39-1	12	56-32-5-10-39-1	11.5	10-5-32-56-39-1	12.5
5	26-40-41-69-27-16-1	11	Misma que la actual	11	Misma que la actual	11
6	25-47-24-14-59-19-1	10	24-47-14-59-25-19-1	7.5	Misma que en AMPL	7.5
7	22-28-68-57-38-1	12	Misma que la actual	12	Misma que la actual	12

Rutas del martes						
Número	Ruta actual	Coste	Ruta AMPL	Coste	Ruta R	Coste
1	17-29-35-23-37-64-1	12.5	Misma que la actual	12.5	Misma que la actual	12.5
2*	4-8-51-11-66-46-1	13.5	4-51-8-11-46-66-1	12.5	4-51-8-11-66-46-1	13
3	65-31-7-36-13-1	14.5	31-65-7-36-13-1	11.5	Misma que en AMPL	11.5
4	62-39-32-5-1	13.5	32-5-62-39-1	9.5	Misma que en AMPL	9.5
5	33-40-63-61-69-16-1	13	33-40-61-63-69-16-1	13	Misma que la actual	13
6	25-12-67-24-58-1	11.5	24-58-67-12-25-1	9.5	Misma que en AMPL	9.5
7	28-53-3-1	9.5	53-3-28-1	9	Misma que en AMPL	9

Rutas del miércoles						
Número	Ruta actual	Coste	Ruta AMPL	Coste	Ruta R	Coste
1*	35-23-42-37-30-13-1	11.5	35-23-42-30-37-13-1	11.5	30-37-42-23-35-13-1	13.5
2	48-49-70-8-51-44-45-11-50-1	18	50-11-45-44-51-8-70-49-48-1	17.5	50-11-51-45-44-8-70-49-48-1	17.5
3	55-31-7-36-9-1	13.5	31-55-7-36-9-1	11	Misma que en AMPL	11
4	60-32-56-5-10-39-1	13.5	56-32-60-10-5-39-1	13.5	Misma que en AMPL	13.5
5	26-40-41-69-27-16-1	11	Misma que la actual	11	Misma que la actual	11
6**	18-24-58-47-14-59-19-1	8.5	24-58-18-47-14-59-19-1	9.5	59-58-24-18-47-14-19-1	9
7*	28-68-57-6-38-1	11	Misma que la actual	11	28-68-57-38-6-1	11.5
8*	25-54-21-43-15-34-1	10	21-43-54-15-34-25-1	8	21-54-43-34-15-25-1	9

Rutas del jueves						
Número	Ruta actual	Coste	Ruta AMPL	Coste	Ruta R	Coste
1	17-29-35-23-37-64-1	12.5	Misma que la actual	12.5	Misma que la actual	12.5
2*	20-8-51-52-11-66-46-1	15	20-51-8-52-11-46-66-1	14.5	20-51-52-8-11-66-46-1	15
3	65-31-7-36-13-1	14.5	31-65-7-36-13-1	11.5	Misma que en AMPL	11.5
4	62-39-1	5	Misma que la actual	5	Misma que la actual	5
5	33-2-40-63-61-69-16-1	13.5	33-2-40-61-63-69-16-1	13.5	Misma que la actual	13.5
6	25-12-67-24-1	10.5	24-67-12-25-1	8.5	Misma que en AMPL	8.5
7	28-53-3-32-5-1	13	53-28-3-32-5-1	13	Misma que la actual	13

Rutas del viernes						
Número	Ruta actual	Coste	Ruta AMPL	Coste	Ruta R	Coste
1	35-23-42-37-30-1	10.5	Misma que la actual	10.5	Misma que la actual	10.5
2	48-49-70-8-51-45-11-50-1	17	50-11-45-51-8-70-49-48-1	16.5	50-11-51-45-8-70-49-48-1	16.5
3	55-9-34-15-1	8.5	55-9-15-34-1	8	55-9-15-34-1	8
4	32-5-10-62-39-1	12	32-10-5-62-39-1	11.5	Misma que en AMPL	11.5
5	40-41-69-27-54-21-43-1	14.5	21-54-43-40-41-69-27-1	13.5	54-43-21-41-40-69-27-1	13.5
6	25-24-58-14-59-19-16-1	12.5	24-58-14-59-16-25-19-1	10	Misma que en AMPL	10
7	28-68-57-6-1	9.5	68-28-57-6-1	9.5	Misma que la actual	9.5

6.2. Discusión de los resultados

Podemos ver como, en general, con los cambios heurísticos aquí aplicados, los resultados en AMPL son mejores y se consiguen de manera más rápida. Para una ruta esto podría ser previsible. Sin embargo, cuando queremos realizar varias rutas a la vez, la búsqueda heurística va a resultar más rápida y, lo más importante, se podrá realizar de forma mucho más ligera computacionalmente. Así, podríamos pensar en realizar los cambios dentro de la misma ruta con ayuda de AMPL, aunque requeriría resolver hasta siete problemas en cada cambio y podría resultar incómodo reescribir los datos para resolver los problemas y luego volver a escribirlos en R con las soluciones encontradas en AMPL. Además, este tipo de ejecución sólo sería recomendable si no tenemos un proceso directo para realizar estos cambios, es decir, que esté por partes, como es nuestro caso actual. Si bien podríamos pensar esto mismo para los cambios con dos rutas, problemas que, en general, podemos resolver, tendremos que esperar al menos cuatro minutos en nuestras condiciones para tener una solución con dos rutas, ya que es de esperar que haya al menos diez nodos en estas rutas.

Si nos centramos en las rutas con un asterísco (*) lo que veremos son aquellas rutas individuales de las cuales, con el proceso actual de cambio de orden de nodos, no alcanzan las soluciones y, por consiguiente, los costes que se obtienen en AMPL con los datos que tenemos. Hay dos explicaciones para que no se alcancen estas soluciones y de las cuales podemos establecer su solución:

- Para las rutas *lunes 4* y *miércoles 1* tenemos lo ya explicado y propuesto para los cambios de los nodos intermedios de una ruta. Al realizar las sustituciones para una ruta llega un momento en el cual un nodo o conjunto de nodos quedan temporalmente aislados del árbol que llega al CHUS. Por defecto, consideramos que este conjunto de nodos debe volver a conectarse por el nodo del cual hemos eliminado el arco. Si solo queda ese nodo aislado no tenemos mayor inconveniente ni capacidad de elección. Sin embargo, en caso de quedar un conjunto de nodos libre, la elección realizada por defecto, que ya se ha explicado anteriormente, puede no ser la mejor opción.

- En el caso de las rutas *martes 2*, *miércoles 7*, *miércoles 8* y *jueves 2* vemos que no alcanzamos costes similares a los de AMPL porque un par de nodos están al revés de como sería ideal, lo que aumenta ligeramente el coste de la ruta. Esto podría solucionarse con un tipo de cambio que no habíamos pensado: el cambio de orientación entre dos nodos intermedios. Esto significa cambiar de orden un par de nodos, manteniendo los nodos anterior y posterior a este par, cambiando su nodo destino y origen, respectivamente.

Por otro lado, tenemos marcada la ruta *miércoles 6* con dos asteríscos (**). Esta es la única ruta en la que la solución de AMPL tiene un coste mayor que la ruta actual y también que la obtenida por R. La razón está en los datos del nodo 18 (Codeseda), pues coincide en la ruta con el nodo 24 (A Estrada) y el tiempo aproximado de llegada de un centro a otro se establece en 0, lo supone un tiempo total de 0,5 unidades de tiempo ir de Codeseda a A Estrada. Sin embargo, este desplazamiento inmediato es, a día de hoy, imposible. A nivel de la programación en AMPL, lo que implica esto es que podrían considerarse los periodos de tiempo de llegada a ambos nodos

iguales, cosa que, si bien no es imposible, al trabajar con las distintas formulaciones en AMPL este programa impide que suceda:

- Con la formulación base: el tiempo total de 0,5 en un nodo distinto del CHUS no lo considera para la solución.
- Con la formulación con redondeos: Los tiempos de las restricciones 8 y 9 acabarán siendo los mismos y no se permite igualar los periodos de tiempo.
- Con la formulación con función suelo y función techo: Sucede lo mismo que en el caso con redondeos. Además, con 0,5 unidades de tiempo, es el único caso en el que sucede. Por lo tanto, en general, esto no debería ser problema para nosotros en un caso real, pero conviene buscar otra formulación en caso de considerar datos de distinta naturaleza, como unidades de tiempo mayores.

De hecho, si ponemos un coste de 0,1 entre estas localidades, la ruta obtenida acaba siendo la actual y comprobamos que la razón del mayor coste en el caso heurístico proviene de no considerar la unión de la ruta con el nodo que inicia esa ruta en la solución inicial.

Capítulo 7

Conclusiones y posible trabajo futuro

En base a los resultados obtenidos y a la manera en la que hemos enfrentado el problema propuesto, podemos realizar los siguientes comentarios:

- Al compararlo con los resultados obtenidos en AMPL, el método heurístico aquí propuesto es bueno en general, al menos en lo que a los cambios dentro de una ruta se refiere.
- Si bien no siempre se obtiene una solución óptima, esto podrá ser debido a que los tipos de cambios aquí propuestos son insuficientes y podríamos añadir alguno más como el cambio de orientación entre nodos intermedios. En otros casos, deberíamos modificar ligeramente los cambios que ya tenemos presentados, dando lugar a más elecciones a la hora de unir de nuevo las rutas y obtener, de esta manera, un coste menor.
- A pesar de que en las tablas de resultados presentadas en el capítulo 6 no han sido necesarios los cambios entre rutas, a nivel de escritura de código estos casos son más sencillos que los anteriores, por lo que podremos suponer que funcionan mejor en cuanto a tiempo de computación.
- Se puede pensar en mejorar los códigos. La primera opción sería la introducción de un criterio para intentar, en la medida de lo posible, que se cumpla el límite establecido de duración de una ruta y recolocar en el árbol los nodos que puedan quedarse aislados. Además, sería recomendable una mejora en los códigos para que queden más limpios, eliminar cosas que puedan resultar redundantes o añadir criterios que separen un cambio intermedio en la misma ruta de uno entre distintas rutas.
- La mejora más importante que se debería realizar es una forma de unificar todos los cambios en un único código, estableciendo criterios de parada adecuados, considerando las excepciones ya explicadas para mantener un cambio sin que éste necesariamente mejore el coste que ya tenemos, etc.
- El algoritmo de solución inicial en este momento no obliga a utilizar todas las rutas deseadas, aunque en general siempre se conseguirá tener el número de rutas deseado. En caso de que esto no sea así, nos quedaría a nuestro criterio ver la manera de solucionarlo o darle significado.

- A nivel de tiempo de computación, para una ruta resulta más lento el método heurístico presentado que el uso de AMPL, si bien en este caso podremos trabajar con más de tres rutas sin problemas, lo cual es una ventaja a tener en cuenta.

En conclusión, en este trabajo se han establecido unas bases para la búsqueda de rutas de ambulancias en el área sanitaria de Santiago de Compostela y Barbanza, las cuales pueden ser extrapoladas a otros conjuntos de rutas de vehículos de distinta índole. Si bien no puede considerarse un método de planificación de rutas definitivo, se han propuesto algunas mejoras para estas bases, las cuales se podrían analizar en un futuro trabajo. Otra tarea que se podría considerar realizar sería ver qué sucede con la inclusión o eliminación de nuevos nodos en distintos días, las rutas que se consiguen en estos casos o, incluso, tratar de mejorar las rutas con las localidades que ya tenemos y comprobar si podemos no sobrepasar en exceso el límite de tiempo en cada ruta.

Este se puede considerar como el inicio de un conjunto de análisis y estudios sobre como mejorar no solo las rutas de transporte tanto en el caso visto aquí como en otros casos análogos, sino también mejorar los métodos con los que buscamos las soluciones a la espera de poder contar con un equipo con la capacidad suficiente como para afrontar nuestro problema en un tiempo razonable, cosa que con un ordenador personal no parece ser, a día de hoy, algo demasiado factible.

Apéndice A

Listado de localizaciones

1	CHUS	19	Conxo	36	Negreira	54	Rois
2	Abella	20	Corrubedo	37	Noia	55	Santa Comba
3	Agolada	21	Dodro	38	Oca	56	Santiso
4	Aguiño	22	Dozón	39	Pino	57	Silleda
5	Arzúa	23	Esteiro	40	Ordes	58	O Souto de Vea
6	Bandeira	24	A Estrada	41	Oroso	59	Os Tilos
7	Baña	25	Fontiñas	42	Outes	60	Toques
8	Barbanza	26	Frades	43	Padrón	61	Tordoia
9	Bertamiráns	27	Galeras	44	Palmeira	62	Touro
10	Boimorto	28	Lalín	45	Pobra do Caramiñal	63	Trazo
11	Boiro	29	Lira	46	Pontecesures	64	Urdilde
12	Boqueixón	30	Lousame	47	Pontevea	65	Val do Ubra
13	Brion	31	Mazaricos	48	Portosín	66	Valga
14	Cacheiras	32	Melide	49	Porto do Son	67	Vedra
15	Calo	33	Mesía	50	Rianxo	68	Vila de Cruces
16	Carenal	34	Milladoiro	51	Ribeira	69	Vite
17	Carnota	35	Muros	52	Ribeira 2	70	Santa Mariña de Xuño
18	Codeseda			53	Rodeiro		

Apéndice B

Archivos modelo en AMPL. Versión para uso en NEOS

routes.mod

```
set N ordered;
set R ordered;
set T ordered;

param C {i in N,j in N}>=0;
param P {i in N}>=0;
param D {v in R}>=0;
param n := card(N);
param r := card(R);
param z1 >=0;
param z2 >=0;

var X{i in N,j in N,t in T,v in R} binary;

minimize Time:
  sum{i in N,j in N,t in T,v in R} (C[i,j]+P[i])*X[i,j,t,v];

subject to Tree:
  sum{i in N,j in N,t in T,v in R} X[i,j,t,v]=n-1;

subject to Hospital {v in R}:
  sum{i in N diff{first(N)},t in T} X[i,1,t,v]=1;

subject to NoCycle {i in N,j in N,l in N,v in R,u in R,t in T,k in T:t>= k}:
```

```
  X[i,j,k,v]+X[l,i,t,u]<=1;

subject to Allpickup {i in N diff{first(N)}}:
  sum{j in N,t in T,v in R} X[i,j,t,v]=1;

subject to Nolink {i in N,t in T,v in R}:
  X[i,i,t,v]=0;

subject to NoRepeat {j in N diff{first(N)}}:
  sum{i in N,t in T,v in R} X[i,j,t,v]<=1;

subject to SameRoute {j in N,i in N diff{j},l in N diff{j},
v in R,u in R diff{v},t in T,k in T}:
  X[i,j,t,v] + X[j,l,k,u]<=1;

subject to TimeBetweenTwoPoints1 {i in N,j in N ,l in N,
v in R,u in R,t in T,k in T diff{t}:t<k+ #Puesto de esta forma para incluir o no
# el redondeo. Análogo para la siguinete restricción
floor(C[i,j]+P[i])}:
  X[i,j,k,v]+X[j,l,t,u]<=1;

subject to NoGap {i in N,j in N,l in N ,
v in R,u in R,t in T,k in T diff{t}:t>k+
2*(P[i]+C[i,j])-ceil(C[i,j]+P[i])}:
  X[i,j,k,v]+X[j,l,t,u]<=1;

subject to StartSoon {v in R}:
  sum{i in N,j in N diff{i},t in first(T)..first(T)+11} X[i,j,t,v]>=1;

subject to IntervalArrive {i in N,j in N diff{i},t in T,k in T,v in R,
u in R diff{v}:t+C[i,1]-C[j,1]-0.5<=k<=t+C[i,1]-C[j,1]+0.5}:
  X[i,1,t,v]+X[j,1,k,u]<=1;

subject to TimeLimit {v in R}:
  sum{j in N,i in N diff{first(N)},t in T} (C[i,j]+P[i])*X[i,j,t,v]<=D[v];
```

routes.dat

```
set N := 1 14 18 19 24 47 58 59
; # Nodos: "CHUS" "CACHEIRAS" "CODESEDA" "CONXO"
# "ESTRADA" "PONTEVEA" "SOUTOVEA" "TILOS"
```

```
set R := 1 #2 3 4 5 6 7 8
; # Número de rutas

set T := 0 1 2 3 4 5 6 7 8 9 10 11 12 13 #14 15 16 17 18 19 20 21 22 23
; # 1=10 min Poner de 0 a 11+(r-1).

param D :=
1 12
#2 12
#3 12
#4 12
#5 12
#6 12
#7 12
#8 12
;

param C: 1 14 18 19 24 47 58 59 :=
1 0 2 3 0.5 3 2 2.5 1.5
14 2 0 2 1 2 0.5 1 1
18 3 2 0 2.5 0.1 1 1 2
19 0.5 1 2.5 0 2.5 1.5 2 1
24 3 2 0.1 2.5 0 1 1 2
47 2 0.5 1 1.5 1 0 1 1
58 2.5 1 1 2 1 1 0 1.5
59 1.5 1 2 1 2 1 1.5 0
; # Tiempos de trayecto entre nodo i y nodo j.

param P:=
1 0
14 0.5
18 0.5
19 0.5
24 0.5
47 0.5
58 0.5
59 0.5
; # Tiempos de carga en el nodo i.
```

routes.run

```
option reset_initial_guesses 1;
option presolve 1;
#option solver gurobi;
option gurobi_options 'presolve=1 mipgap=0.01 nodefilestart=3 timelim=3600';

#problem Ruta: Time,Tree,Hospital,NoCycle,Allpickup,Nolink,NoRepeat,SameRoute,
#TimeBetweenTwoPoints1,NoGap,StartSoon,TimeLimit;

printf "\n Solution:" > solRoutes.txt;
solve #Ruta
;
printf "\n Time: %3.2f \n",Time >> solRoutes.txt;
printf "\n Route design: \n " >> solRoutes.txt;
let z1 := 0;
let z2 := 0;
for {v in R}{
  let tf[v] := min{i in N,j in N,t in T:X[i,j,t,v]=1} Tf[t];
}
for {v in R}{
  for {t in T}{
    for {i in N}{
      for {j in N}{
        if X[i,j,t,v]=1 then {
          let z2 := z2+(C[i,j]+P[i])*X[i,j,t,v];
          printf"\n Start %3i \n",i >> solRoutes.txt;
          printf" End %3i \n",j >> solRoutes.txt;
          printf" Instant %3i \n",t >> solRoutes.txt;
          printf" Real route initial time %3.2f \n", z1 >> solRoutes.txt;
          printf" Real route final time %3.2f \n", z2 >> solRoutes.txt;
          printf" Route %3i \n",v >> solRoutes.txt;
          let z1 := z1+(C[i,j]+P[i])*X[i,j,t,v];
        }
      }
    }
  }
  let z1 := 0;
  let z2 := 0;
}

option presolve_warnings 10;
option omit_zero_rows 1;
```

```
option show_stats 1;
display _solve_elapsed_time ;
display X;
display Time;
for {v in R}{
  for {t in T}{
    for {i in N}{
      for {j in N}{
        if X[i,j,t,v]=1 then {
          printf "(%2i,%2i,%2i,%1i) \n", i, j, t, v;
          display C[i,j]+P[i];
        }
      }
    }
  }
};
```

Apéndice C

Algoritmo de solución inicial en R

```
# Insertar directorio con los datos correspondientes
C=as.matrix(read.table("nuevosdatos.txt")) # Matriz de tiempo de distancias
P=as.matrix(read.table("nuevosdatos2.txt")) # Vector de tiempo de carga
Loc=as.matrix(read.table("nuevosdatos3.txt")) # Vector con los nombres
# de las localidades.
loc=as.matrix(read.table("nuevosdatos4.txt")) # Vector con los números globales
# de las localidades.
C=as.matrix(C[-1,-1]) # Elimina los valores que representan los nodos.
rownames(C)=Loc # Nombre filas y columnas.
colnames(C)=Loc
n=nrow(C) # Número de nodos con el que trabajaremos.
P=P[,-1] # Elimina los valores que representan los nodos.
names(P)=Loc
for (i in 1:n){ # Sumamos los tiempos para tener un coste total.
C[,i]=C[,i]+P[i]
}
C=t(C) # Hacemos que se represente el ir del nodo fila al nodo columna.

rut=1 # Número de rutas que queremos.
k=0 # Fila de arcos en la matriz actual.
r=0 # Última ruta en añadirse a la solución.
arcospro=matrix(0,nrow=n,ncol=3) # Matriz provisional con el arco de menor distancia
# con los criterios explicados en el capítulo 5.
arcos=matrix(0,nrow=n-1,ncol=4) # Matriz de arcos con la solución.
lista1=seq(1:n) # Nodos que no son considerados de salida.
# 1 estará si no tiene tantos arcos como rutas queremos.
lista2=seq(1:n) # Nodos que aún no tienen todas las conexiones posibles.
# Al final, nos indica los nodos que empiezan rutas.
```

```
chus=numeric(n-1)
deschus=numeric(n-1)
ci=0
cambio=numeric(n-1)

while (sum(arcos[,c(1:2)]==0)!=0){ # Criterio de parada cuando no queden
# más arcos por añadirse.
  for (i in lista1){
  x=100
    if (i==1){ # Evitamos que el nodo 1 repita conexión con un nodo ya establecido
      for (j in lista2){
        if (sum(arcos[which(arcos[,c(1:2)]==1)])!=0){
          for (i2 in 1:(n-1)){
            chus[i2]=sum(arcos[i2,c(1:2)]==1)
          }
          for (l in 1:2){
            if (sum(arcos[,l]==1)!=0){
              deschus=arcos[which(chus==1),c(1:2)[-l]]
            }
          }
          if (sum(arcos[,1]==j)>0){
            if (arcos[which(arcos[,1]==j),4]!=0){
              next
            }
          }
          if (i!=j){
            if (C[i,j]<x){ # Tampoco permitimos que un nodo con ruta
                           # ya establecida se pueda unir
              if (sum(deschus==j)!=0){
                next
              }
              else{
                x=C[i,j]
                arcospro[i,1]=i
                arcospro[i,2]=j
                arcospro[i,3]=x
              }
              }
            }
        }
        else{
```

```
      if (i!=j){
        if (C[i,j]<x){
          x=C[i,j]
          arcospro[i,1]=i
          arcospro[i,2]=j
          arcospro[i,3]=x
        }
      }
    }
  }
}
else{# En el caso general:
  for (j in lista2){
    if (i!=j){
      if (sum(arcos[,1]==j)>0){
        if (sum(arcos[,2]==i)>0){
          if (arcos[which(arcos[,1]==j),2]==i){ # No permitimos la orientación
                                                # opuesta de un arco ya establecido
            next
          }
        }
      }
      if (sum(arcos[,1]==i)>0){
        if (j==1 & arcos[which(arcos[,1]==i),4]!=0){ # Si el nodo ya tiene una ruta
                                                     # establecida, no podemos
                                                     # conectarlo con 1 otra vez
          next
        }
      }
      if (C[i,j]<x){
        if (sum(deschus==i)!=0 & j==1){ # No volvemos a considerar
                                        # los arcos ya unidos a 1
          next
        }
        else if (sum(arcos[,c(1:2)]==i)>0 & sum(arcos[,c(1:2)]==j)>0){
          if (sum(arcos[which(arcos[,1]==i),4]!=0)>0
              & sum(arcos[which(arcos[,1]==j),4]!=0)>0){
            if (arcos[which(arcos[,1]==i),4]!=arcos[which(arcos[,1]==j),4]){
              next # No se unen nodos con rutas definidas distintas
            }
            else {
```

```
                x=C[i,j]
                arcospro[i,1]=i
                arcospro[i,2]=j
                arcospro[i,3]=x
              }
            }
            else {
              x=C[i,j]
              arcospro[i,1]=i
              arcospro[i,2]=j
              arcospro[i,3]=x
            }
          }
          else{
            x=C[i,j]
            arcospro[i,1]=i
            arcospro[i,2]=j
            arcospro[i,3]=x
          }
        }
      }
     }
  }
}
# arcospro # Arcos con las condiciones deseadas para analizar cuales
           # se incluyen en la solución.
newnodos=0 # Arcos que incluimos en esta iteración.
for (i in 1:n){
  for (j in 1:n){
    if (arcospro[i,1]==arcospro[j,2] &
        arcospro[j,1]==arcospro[i,2] & arcospro[i,1]!=0){ # Mismo arco de coste
                                                          # mínimo entre nodos
                                                          # en ambos sentidos

      if (sum(arcos!=0)!=0){
        if (arcos[k,1]==arcospro[i,1] &
            arcos[k,2]==arcospro[i,2]){ # Si el arco ya está en la matriz,
          next
        }
      }
      k=k+1
```

```
      newnodos=newnodos+1
      if (k>(n-1)){ # Aseguramos la cantidad de arcos deseada
        k=n-1
      }
      if (arcospro[j,2]==1){ # Si el nodo de llegada es el 1, tenemos una nueva ruta
        r=r+1
        arcos[k,1]=arcospro[j,1]
        arcos[k,2]=arcospro[j,2]
        arcos[k,3]=arcospro[j,3]
        arcos[k,4]=r
      }
      else if (arcospro[j,1]==1){ # No incluimos el nodo 1 como nodo de salida
        #print('Nodo 1 solo de llegada')
        k=k-1
        newnodos=newnodos-1
        next
      }
      else { # Incluimos un arco con llegada distinta al nodo 1
        arcos[k,1]=arcospro[i,1]
        arcos[k,2]=arcospro[i,2]
        arcos[k,3]=arcospro[i,3]
      }
      if (sum(which(arcos[k,1]==arcos[,2]))!=0 &
          sum(which(arcos[k,2]==arcos[,1]))!=0){
      # Si el arco ya está en la lista, no lo volvemos a poner
        #print(paste('Nodos',i,'y',j,'ya en la lista de arcos'))
        arcos[k,]=0
        k=k-1
        newnodos=newnodos-1
      }
    }
  }
}
r1=numeric(4) # Reservamos arcos para cambiarlos
r2=numeric(4)
if (r>0){ # En caso de añadir un nodo con llegada a 1, se lleva el arco a la fila
          # en posición a la ruta que indica este arco
  if (arcos[r,2]!=1){
    if (newnodos!=1){
      r1=arcos[r,]
      r2=arcos[k-newnodos+1,]
```

```
      arcos[k-newnodos+1,]=arcos[k,]
      arcos[r,]=r2
      arcos[k,]=r1
    }
    if (newnodos==1){
      r1=arcos[r,]
      arcos[r,]=arcos[k,]
      arcos[k,]=r1
    }
  }
}
arcos2=arcos # Reservamos lo ya hecho por si necesitasemos recuperarlo
li=0
m=numeric(2) # Reserva de arcos para cambiar la orientación
lista1=numeric(n) # Lista de nodos que no consideraremos de llegada
                  # para la próxima iteración
if (sum(arcos[,c(1:2)]==1)==rut){ # Si ya tenemos todas las rutas deseadas,
                                  # no necesitamos este nodo para comparar costes
  li=1
  lista1[li]=1
}
li2=li # Reservamos si tenemos o no el nodo 1 descartado en caso de posibles cambios
ci=0
cambio=numeric(n-1)
i=1
while (i<=n){ # Cambios de orientación de los nodos para evitar considerar
              # el mismo nodo solo de llegada o solo de salida
  i=i+1
  if (ci>3*n){ # Criterio de parada extra
    print ('Revisar cambios')
    break
  }
  if (sum(arcos[,c(1:2)]==i)>1){ # Nodo i con más de una aparición
    if (sum(arcos[,1]==i)>1){ # Repetido en la columna 1
      m=which(arcos[,1]==i) # Filas en las que están esos arcos
      if (sum(cambio==m[2])>0 &
        arcos[m[1],2]==1){ # Si el posible cambio se hace con una llegada a 1
                           # y se ha cambiado el arco de abajo, paramos
        arcos=arcos2
        li=li2
        ci=0
```

```
        cambio=numeric(n-1)
        i=1
        break
      }
      else if ((sum(cambio==m[2])>0 & arcos[m[1],2]!=1)){ # Si se ha cambiado el
                                                          # arco de abajo, se cambia
                                                          # el de arriba sin nodo 1
        p=arcos[m[1],2]
        arcos[m[1],2]=arcos[m[1],1]
        arcos[m[1],1]=p
        ci=ci+1
        cambio[ci]=m[1]
        i=1
      }
      else{ # Por defecto, cambio en la fila de abajo
        p=arcos[m[2],2]
        arcos[m[2],2]=arcos[m[2],1]
        arcos[m[2],1]=p
        ci=ci+1
        cambio[ci]=m[2]
        i=1
      }
    }
    if (sum(lista1==i)==0){ # Lista de nodos considerados de salida.
      li=li+1
      lista1[li]=i
    }
  }
}
if (length(which(arcos==arcos2))==4*(n-1)){ # Si se llega a un cambio del nodo 1,
                                            # tomamos como predeterminado
                                            # el cambio del nodo de arriba
  while (i<=n){
    i=i+1
    if (ci>3*n){
      print ('Revisar cambios')
      break
    }
    if (sum(arcos[,c(1:2)]==i)>1){
      if (sum(arcos[,1]==i)>1){
        m=which(arcos[,1]==i)
```

```
        if ((sum(cambio==m[1])>0 | arcos[m[1],2]==1)){
          p=arcos[m[2],2]
          arcos[m[2],2]=arcos[m[2],1]
          arcos[m[2],1]=p
          ci=ci+1
          cambio[ci]=m[2]
          i=1
        }
        else{
          p=arcos[m[1],2]
          arcos[m[1],2]=arcos[m[1],1]
          arcos[m[1],1]=p
          ci=ci+1
          cambio[ci]=m[1]
          i=1
         }
      }
      if (sum(lista1==i)==0){
        li=li+1
        lista1[li]=i
      }
    }
  }
}
lista1=seq(1:n)[-lista1] # Nos quedamos con los nodos que no son de llegada
if (li==0){ # Serán todos en caso de solo haber incluido un arco de llegada a 1
            # y con rutas disponibles
  lista1=seq(1:n)
}
if ((sum(arcos[,2]==1))<rut){ # Si hay rutas disponibles,
                              # se incluye el nodo 1 para ver el arco a incluir
  lista1=c(1,lista1)
}
for (l in 1:2){ # Nodos ya en la solución
  lista2=sort(arcos[which(arcos[,l]!=0),l])
}
lista2=seq(1:n)[-lista2] # Nos quedamos con aquellos que no son de llegada
if (sum(arcos[,c(1:2)]==1)<rut){ # Si no se llega a 1 tantas veces como rutas hay,
                                 # incluimos el 1 en este conjunto
  lista2=c(1,lista2)
}
```

```
arcospro=matrix(0,nrow=n,ncol=3) # Reiniciamos los arcos de
                                 # menor coste en cada nodo
nodact=numeric(sum(arcos[,1]!=0)) # Vector con los nodos
                                  # distintos a 1 en la solución
col2=1
nuevo=col2
i=1
while (i<=length(nodact)){ # Buscamos los nodos que tienen
                           # alguna conexión con el nodo 1
                           # para definir su ruta
  if (nuevo==1){ # El nodo que acompaña al 1 ya tiene ruta
    nodact[i]=arcos[col2,1]
    nuevo=arcos[col2,1]
    i=i+1
  }
  else if (sum(arcos[,2]==nuevo)==0){ # El nodo considerado no es de llegada
    if (col2<sum(arcos[,2]==1)){ # Si no hemos recorrido todas las rutas
                                 # establecidas, pasamos al siguiente
      col2=col2+1
      nuevo=1
    }
    else { # Si las rutas se han recorrido, acabamos el proceso
      i=length(nodact)+1
    }
  }
  else{ # Incluimos los nodos que acompañan al último nodo incluido
    nodact[i]=arcos[which(arcos[,2]==nuevo),1]
    nuevo=arcos[which(arcos[,2]==nuevo),1]
    i=i+1
  }
}
if (sum(nodact==0)!=0){ # Si hay nodos no considerados, nos quedamos
                        # solo con aquellos elementos distintos a 0
  nodact=nodact[-which(nodact==0)]
}
for (i in 1:length(nodact)){ # Redefinimos las rutas con los nodos
                             # que ya tienen ruta
  if (arcos[which(arcos[,1]==nodact[i]),4]!=0){
    arcos[which(arcos[,2]==nodact[i]),4]=arcos[which(arcos[,1]==nodact[i]),4]
  }
}
```

```
}
arcos # Presentación de la solución
(total=sum(arcos[,3])) # Coste de nuestra solución

for (l in 1:2){
  if (sum(arcos[,l]==1)>0){
    nodos1=arcos[,l]
    nodos2=arcos[,c(1:2)[-l]]
  }
}


lista2=numeric(rut) # Reservamos este vector para los nodos que inician rutas
for (i in 2:n){
  if (sum(arcos[,c(1:2)]==i)==1){
    lista2[arcos[arcos[,1]==i,4]]=i
  }
}
r=max(arcos[,4]) # Cantidad de rutas en la solución presentada
while (r<rut){ # Ampliamos hasta obtener el número de rutas deseado
               # con procesos análogos a los ya vistos
  if (sum(lista2==0)>0){
    lista2=lista2[-which(lista2==0)]
  }
  r=r+1
  x=100
  for (i in 1:length(lista2)){
    if (arcos[which(arcos[,1]==lista2[i]),2]!=1){
      if (x>C[lista2[i],1]){
        x=C[lista2[i],1]
        nl2=lista2[i]
      }
    }
  }
  arcos[which(arcos[,1]==nl2),1]=nl2
  arcos[which(arcos[,1]==nl2),2]=1
  arcos[which(arcos[,1]==nl2),3]=x
  arcos[which(arcos[,1]==nl2),4]=r
  lista2=numeric(rut)
  for (i in 2:n){ # Actualizamos la lista de nodos iniciales
    if (sum(arcos[,c(1:2)]==i)==1){
```

```
      lista2[arcos[arcos[,1]==i,4]]=i
    }
  }
}
(arcos=arcos[order(arcos[,2],arcos[,4]),]) # Ordenamos los arcos por número de nodo
                                           # y por ruta en caso del nodo 1.
tiemporutas=numeric(rut)
for (i in 1:rut){ # Para el cálculo de coste de cada ruta
  tiemporutas[i]=sum(arcos[which(arcos[,4]==i),3])
}
tiemporutas # Coste de cada ruta
```

Bibliografía

[1] Anaya-Arenas AM, Chabot T, Renaud J, Ruiz A (2016) Biomedical sample transportation in the province of Quebec: A case study. International Journal Of Production Research54:602-615.

[2] Anaya-Arenas AM, Prodhon T, Renaud J, Ruiz A (2019) An iterated local search for the biomedical sample transportation problem with multiple and interdependent pickups. Document de travail. CIRRELT, Centre Interuniversitaire de Recherche sur les Réseaux d'Enterprise, la Logistique et le Transport. Faculté des Sciences de l'Administration de l'Université Laval, Canada.

[3] Casas-Méndez B, Davila-Pena L (2021a) Problemas de transporte de mostras biomédicas. Boletín INFORMEST-SGAPEIO 59:2-4.

[4] Casas-Méndez B, Davila-Pena L. (2021b) Correcting blood potassium discrepancies: Optimization of the ambulance routes of a clinical analysis service. Documento de trabajo.

[5] Chandy KM, Lo T (1973) The capacitated minimum spanning tree. Networks 3:173-181.

[6] Clarke G, Wright JR (1964) Scheduling of vehicle routing problem from a central depot to a number of delivery points. Operations Research, 12:568-581.

[7] Coltin B, Veloso M (2014) Scheduling for transfers in pickup and delivery problems with very large neighborhood search. Proceedings of the National Conference on Artificial Intelligence 3:2250-2256.

[8] Dantzig GB, Ramser JH (1959) The truck dispatching problem. Management Science 6:80–91.

[9] Espasandín-Domínguez J, Benítez-Estévez AJ, Cadarso-Suárez C, Kneib T, Barreiro-Martínez T, Casas-Méndez B, Gude F (2018) Geographical differences in blood potassium detected using a structured additive distributional regression model. Spatial Statistics 24:1-13.

[10] Kergosien Y, Lenté C, Billaut JC, Perrin S (2013) Metaheuristic algorithms for solving two interconnected vehicle routing problems in a hospital complex. Computers & Operations Research 40:2508–2518.

[11] Miller CE, Tucker AW, Zemlin RA (1960) Integer problems. Journal of Association for Computing Machinery 7:326–329.

[12] Mobasher A, Ekici A, Ozener O (2015) Coordinating collection and appointment scheduling operations at the blood donation sites. Computers and Industrial Engineering 87:260-266.

[13] Naji-Azimi Z, Salari M, Renaud J, Ruiz A (2016) A practical vehicle routing problem with desynchronized arrivals to depot. European Journal of Operational Research 255:58–67.

[14] Parragh SN, Doerner KF, Hartl, RF (2008a) A survey on pickup and delivery problems. Journal Für Betriebswirtschaft 58:21–51.

[15] Parragh SN, Doerner KF, Hartl, RF (2008b) A survey on pickup and delivery problems. Journal Für Betriebswirtschaft 58:81–117.

[16] Solomon MM (1986) The minimum spanning tree problem with time window constraints. American Journal Of Mathematical And Management Sciences 6:399-421.

[17] Yi J (2003) Vehicle routing with time windows and time-dependent rewards: A Problem from the American Red Cross. Manufacturing & Service Operations Management 5:74–77.

[18] Yücel E, Salman FS, Gel ES, Örmeci EL, Gel A (2013) Optimizing specimen collection for processing in clinical testing laboratories. European Journal of Operational Research 227:503-514.