UNIVERSIDADE
DE VIGO

Automatización de horarios como un problema de flujo en redes.

Francisco de Asís López Álvarez

Máster en Técnicas Estadísticas

Universidade de Vigo

Automatización de horarios como un problema de flujo en redes.

Francisco de Asís López Álvarez

Autorización de entrega

D. Julio González Díaz

Certifica

Que el proyecto titulado “Automatización de horarios como un problema de flujo en redes.” ha sido realizado por D. Francisco de Asís López Álvarez, con D.N.I. 42855509-T, bajo la dirección de D. Julio González Díaz.

Esta memoria constituye la documentación que, bajo mi autorización, entrega dicho alumno como Proyecto Fin de Máster.

Firmado

D. Julio González Díaz

Vigo, a 30 de Junio de 2014

Resumen

Este trabajo presenta un método de resolución para la automatización de horarios considerándolo como un problema de flujo en redes. El objetivo será obtener un programa que asigne automáticamente el horario de los monitores de una instalación deportiva cumpliendo un conjunto de restricciones derivadas del funcionamiento y características específicas de la propia instalación. El enfoque del trabajo será totalmente ingenieril, centrándose en obtener una solución óptima al problema que contente a todas las partes implicadas. El programa fue creado, adaptado y probado para un caso real, concretamente el club Metropolitan de Vigo. En la presente memoria se muestran los avances obtenidos hasta el 1 de Junio de 2014, ya que actualmente el programa sigue evolucionando para adaptarse a nuevas exigencias y características especiales de otro tipo de instalaciones.

Índice general

Capítulo 1

Introducción.

El presente trabajo final de máster propone un método para la automatización de horarios aplicado a una instalación deportiva, concretamente al gimnasio Metropolitan de Vigo, no obstante, el método podría adaptarse a la optimización de horarios de otro tipo de instalaciones como colegios, universidades o empresas.

La idea inicial de realizar este trabajo surgió como usuario de la instalación, al detectar ciertos fallos en la coordinación del centro. Hablando con algún monitor, se descubrió la dificultad que entrañaba la asignación de monitores a cada actividad, debido a la cantidad de monitores, actividades diferentes y al conjunto de condiciones especiales que se deberían cumplir. Interesados en conocer y profundizar en el problema se organizó una reunión con la dirección del centro, donde conocimos todas las condiciones a tener en cuenta, sorprendiéndonos que dicho trabajo era realizado a mano por el coordinador, ocupándole gran parte de su jornada laboral, especialmente a la hora confeccionar el horario vacacional.

Este tipo de problemas de automatización de horarios, conocido en la literatura como *"automated timetabling"*, ha tenido en los últimos cincuenta años una considerable atención, iniciados por Gotlied en 1963. Un gran número de variantes de los problemas de horarios han sido planteados en la literatura dependiendo del tipo de institución involucrada y de los tipos de restricciones consideradas, nuestro problema podría clasificarse dentro de los llamados *"School timetabling"* en los que se trata de programar las clases de un colegio asignando profesores a clases, evitando que coincidan dos profesores a la vez en la misma clase y viceversa.

En la extensa literatura referente a este problema, existen multitud de enfoques para obtener la solución, A. Schaerf en "A Survey of Automated Timetabling (1999)" o E.K. Burke (2008) analizan las distintas soluciones empleadas a lo largo de la historia.

Alguna de las soluciones más empleadas para resolver este tipo de problema son los modelos combinatorios que consisten en evaluar cada posible solución, o métodos de programación entera mixta que de alguna forma buscan la solución sin evaluar todo el conjunto de posibles soluciones. Sin embargo, presentan una importante complejidad computacional puesto que incluso un problema de tamaño pequeño tendría un número elevado de posibles soluciones, por lo que se hace necesario buscar otros métodos de solución más eficientes.

En el presente trabajo se propone interpretar el problema de automatización de horarios como un problema de programación lineal, concretamente como un problema de flujo a coste mínimo (PFCM). Osterman y D. de Werra (1983) redujeron el problema de los horarios a una sucesión de problemas de flujo, creando una red para cada periodo de tiempo y D. de Werra (1985) propuso un método similar creando una red para cada clase. La construcción de la red se repite para todas las clases y si se encuentra una solución para todas las redes, se obtiene un calendario completo. Obviamente, ya que no hay retroceso en las clases programadas anteriormente, no se tiene garantía de que la solución se encuentre siempre.

Siguiendo la misma idea, se pretende construir una única red, que incluya todos los profesores, clases y periodos, que en nuestro caso serán monitores, actividades y horas. Este enfoque aunque complica el diseño de la red, nos asegura obtener siempre una solución factible. Además, el considerarlo como un PFCM permite resolver problemas relativamente grandes en un tiempo muy reducido, como se podrá comprobar en el desarrollo del presente documento.

Un problema de flujo a coste mínimo es un tipo de problema de optimización en redes. En rasgos generales diremos que se trata de determinar por donde ha de circular el flujo en la red para que se desplacen las unidades necesarias al menor coste. Imaginemos un problema del transporte, que es un tipo de PFCM, planteado y resuelto por Hitchcock (1941) con anterioridad a la formulación del concepto general de la programación lineal y resuelto por Dantzig, G. B. (1951) como un problema de programación lineal (método Simplex).

El objetivo de los modelos de transporte es encontrar la solución a un coste mínimo para la realización de un plan de envíos, transporte o distribución, desde cualquier grupo de centros de abastecimiento o suministro llamados orígenes a cualquier grupo de centros de recepción llamados destinos. Es decir, determinar la cantidad de productos o mercancías que deben enviarse desde cada punto de origen a cada punto de destino, teniendo en cuenta las restricciones propias del problema referidas a las capacidades o disponibilidades de los centros de abastecimiento y las demandas de los centros de destino, de manera que se minimicen los costes totales de transporte o distribución.

En nuestro caso, el problema trataría de asignar a cada uno de los monitores de la instalación el horario semanal de actividades a realizar, teniendo en cuenta que cada uno de ellos tiene una preferencia horaria diferente y un tipo de actividades concretas a desempeñar. Pues bien, lo que se pretende es adaptar nuestro problema a un PFCM en donde las unidades que circularan por la red sean horas y los costes de distribución las preferencias de los monitores.

En resumen, el objetivo principal de este trabajo será diseñar un programa que asigne automáticamente a cada monitor las horas que debe impartir en la instalación, de forma que no se comentan errores en dicha asignación y a su vez se cumplan sus preferencias. De esta forma aumentará el bienestar general, que sin duda, se verá reflejado en un mejor funcionamiento de la instalación.

El poder automatizar el método de asignación, permitirá una mayor flexibilidad de la instalación, sobre todo a la hora de adaptarse a un cambio en el horario de actividades del centro, a la repentina baja de uno de los monitores o en la elaboración del horario en periodo vacacional, cuestión que hoy en día provoca un trastorno importante al coordinador de la instalación.

Para poder desarrollar el programa fue necesaria la colaboración tanto de la dirección, que nos facilitó toda la información necesaria, del coordinador, que nos aportó todas las condiciones que deberían de cumplir las asignaciones, así como la inestimable colaboración de los monitores del centro que se volcaron en ayudar para obtener el mejor resultado posible. Por todo ello, agradecer a los integrantes del gimnasio Metropolitan de Vigo, su colaboración e interés en el presente proyecto.

Capítulo 2

Características del problema.

Para hacerse una idea general del problema al que nos enfrentamos, pasaremos en primer lugar a describir el funcionamiento del gimnasio y las condiciones especiales de la instalación, que serán imprescindibles para obtener una solución adecuada al problema que nos ocupa.

2.1. Funcionamiento del gimnasio.

El club Metropolitan se encuentra ubicado en el centro comercial *A Laxe* del puerto de Vigo, con 4500 m^2 de instalaciones. Dispone de piscina, salas de fitness, cycling y actividades.

La instalación cuenta con 23 monitores que deben cubrir una franja horaria de apertura de la instalación de 16 horas diarias de lunes a viernes. Los fines de semana son cubiertos por otro grupo de monitores con un conjunto de actividades a impartir muy inferior, con lo que en este caso nos centraremos exclusivamente en la asignación semanal de lunes a viernes.

En el centro existen un total de 18 actividades diferentes, las cuales no pueden ser impartidas por todos los monitores ya que cada uno de ellos tiene sus especialidades, existiendo patrones muy diferentes entre ellos. Alguno de los monitores no imparten ninguna actividad y otros pueden impartir hasta 12 actividades diferentes.

Dentro de las 18 actividades consideradas cabe destacar dos de ellas: *"sala"* y *"piscina"* que a partir de ahora denominaremos como *"sala"* que tendrán un tratamiento

diferente a las demás, esto es debido a que cada monitor por contrato tiene que cubrir cierto número de horas semanales de *"sala"* y de *"actividades"*, en este segundo grupo se incluirán el resto de actividades que se imparten en la instalación.

Para tener una idea de la variedad de tipos de contrato en la Figura 2.1 se muestra la cantidad de horas de *"sala"* y *"actividades"* que tienen que impartir por contrato algunos de los monitores.

		SALA	ACT.
MONITOR	1	20	10
	2	20	10
	3	20	3
	4	20	2
	5	15	7
	6	20	3
	7	20	2
	8	30	0
	9	16	3
	10	15	8

Fig. 2.1: Horas de sala y actividades de alguno de los monitores.

La falta de flexibilidad en estas horas añade un problema extra a la obtención de una solución que contente a todos los monitores y al planteamiento del problema como un PFCM, asunto que trataremos en el Capítulo 4.

Cabe destacar que las actividades dirigidas, "Cycling", "Met Pump" o "Aero Step", constituyen un gasto energético elevado al monitor que las imparte, por ello, será necesario controlar que un monitor no imparta más de cierto número de horas consecutivas de este tipo de actividades.

Otro aspecto a tener en cuenta es que los monitores además de las horas que tienen por contrato pueden realizar en la instalación entrenamientos personales que en muchos de los casos conforma el mayor número de horas impartidas. Aunque estas horas no entran en la designación realizada por el coordinador y son los propios monitores que las gestionan según sus preferencias, afectan indirectamente al problema, ya que muchos de ellos prefieren horarios partidos o con muchas horas libres en el medio de la jornada de trabajo, justo lo contrario que cabría pensar inicialmente antes de

conocer estos detalles del funcionamiento del centro.

Por todo ello, el principal problema al que se enfrenta el coordinador del centro es la realización de los horarios en el periodo vacacional ya que cambian las horas y el número de actividades en la instalación y los monitores toman vacaciones alternativamente, provocando que se tenga que realizar un horario diferente cada semana. Actualmente esta labor ocupa dos meses de trabajo al coordinador y ademas, crea descontento en los monitores ya que sus horarios cambian semanalmente.

Como se ha podido ver hasta ahora, el problema en cuestión tiene un gran número de variables y condiciones, que tendremos que adecuar para poder ponerlo como un PFCM, no obstante, si intentásemos resolver este problema con variables binarias resultaría muy costoso computacionalmente hablando. Por ello, a lo largo del desarrollo del programa aparecerán diferentes cuestiones que podrían resolverse con variables binarias de una forma conceptualmente sencilla. Debido a las dimensiones del problema se han intentado evitar, desarrollando métodos alternativos que aunque más laboriosos de programar obtienen una solución adecuada en pocos segundos.

2.2. Objetivos.

El objetivo del presente trabajo será realizar un programa que asigne automáticamente a cada monitor las actividades que debe impartir, asegurando que:

- Todas las actividades queden asignadas.
- Cada monitor imparta una actividad a la hora y viceversa, es decir, que cada actividad horaria sea impartida por un solo monitor.
- Cada monitor imparta el número de horas de sala y actividades semanales estipuladas por contrato.
- Los monitores no sobrepasen cierto número de horas diarias de trabajo.
- Un monitor no imparta más de dos horas consecutivas de las actividades consideradas como alto gasto energético.
- Adaptarse lo máximo posible a las preferencias de los monitores, tanto horarias como de actividades.

Por otro lado, sera necesario crear un modo vacacional en el programa que asigne las horas a los monitores manteniendo los mismos criterios empleados en el horario anual.

Capítulo 3

El problema de flujo en redes a coste mínimo (PFCM).

En este apartado se explicarán brevemente los problemas de flujo en redes y más concretamente los problemas de flujo a coste mínimo; imprescindible para poder entender como adecuar el problema en cuestión a este tipo de problemas de optimización en redes. La mayor parte del contenido de este capítulo está obtenido de los apuntes de la asignatura "Programación lineal y entera" del profesor Julio González Díaz.

3.1. Redes de flujo.

Un grafo G es un par (N, M) consistente en un conjunto N de elementos llamados nodos o vértices y un conjunto M cuyos elementos son subconjuntos de dos elementos de N, que llamaremos arcos. Además, usaremos n y m para referirnos al número total de nodos y arcos respectivamente.

Un grafo orientado o dirigido es aquel en el que los arcos son pares ordenados; el arco (i, j) empieza en el nodo i y termina en el nodo j. En este caso $M \subseteq N \times N$ y, si $i \neq j$, $(i, j) \neq (j, i)$. En un grafo no orientado, (i, j) y (j, i) representan el mismo arco.

Más adelante será preciso expresar problemas de optimización en grafos como problemas de programación lineal, para ello es necesario resumir en matrices la información relativa a los grafos. Sea $G = (N, M)$ un grafo orientado con nodos dados por $N = \{1, 2, ..., n\}$ y arcos por $M = \{a_1, a_2, ..., a_m\}$. El grafo G puede ser representado por la denominada matriz de adyacencia $\bar{A}_{n\times n}$ definida por:

$$\bar{a}_{ij} = \begin{cases} 1 & \text{si } (i,j) \text{ es un arco de } G \\ \\ 0 & \text{en otro caso} \end{cases}$$

Otra posible representación matricial del grafo G es a través de la matriz de incidencia $\bar{B}_{n\times m}$ definida como:

$$\bar{b}_{ik} = \begin{cases} 1 & \text{si } i \text{ es el nodo inicial } a_k \\ -1 & \text{si } i \text{ es el nodo final } a_k \\ 0 & \text{en otro caso} \end{cases}$$

Una red es un grafo con uno o más números asociados con cada arco o nodo. Estos números pueden representar distancias, costes, fiabilidades u otros parámetros de interés. Llamaremos flujo al envío de elementos u objetos de un lugar a otro dentro de una red. Por ejemplo, el transporte de productos del fabricante al distribuidor, el traslado de las personas desde su domicilio a su lugar de trabajo o en nuestro caso monitores a actividades a impartir. Los modelos correspondientes a estas situaciones los llamaremos modelos de redes con flujo. Por ejemplo, podríamos estar interesados en maximizar la cantidad de un producto transportada de un lugar a otro, o minimizar el coste de envío de un número de objetos desde su lugar de fabricación hasta su destino.

Los objetos que "viajan" o fluyen por la red se llaman unidades de flujo. Las unidades de flujo pueden ser bienes, personas, cartas, información o casi cualquier cosa (el agua que fluye por una red de tuberías, el tráfico que fluye por una red de calles, o en nuestro caso horas).

Cada arco en la red tiene un flujo orientado como nos indicará la flecha del arco. Llamaremos f_k al flujo que pasa por el arco k. A cada arco k le asignamos tres parámetros:

Cota inferior $l_k \geq 0$; Cantidad mínima de flujo que debe pasar por el arco k.

Capacidad o cota superior $u_k \geq 0$; Cantidad máxima de flujo que el arco k puede soportar.

Coste o beneficio c_k; Si es positivo denota el coste unidad de flujo que pasa por el arco k; si es negativo representa beneficios.

En algunos modelos de redes con flujo puede ocurrir que por algunos nodos entre o salga flujo de la red; es lo que llamamos flujos externos. Si es una cantidad fija tendremos un flujo externo fijo (positivo si entra en la red y negativo si sale de la red) y si es una cantidad variable tendremos un flujo externo de holgura. Habrá entonces los siguientes parámetros asociados a cada nodo i:

Cota inferior l_i^e; Si es positivo denota la mínima cantidad de flujo que entra en la red por el nodo i y si es negativo la máxima que sale.

Capacidad o cota superior u_i^e; Si es positivo denota la cantidad máxima de flujo que puede entrar en la red por el nodo i y si es negativo la mínima que tiene que salir.

Coste o beneficio c_i^e; Si es positivo denota el coste unidad de flujo que circula por el nodo i; si es negativo denota beneficio.

Naturalmente, siempre tendremos que $l_i^e \leq u_i^e$. Además, todo nodo será de entrada o de salida de flujo, es decir, l_i^e y u_i^e son los dos no negativos o los dos no positivos (pueden ser los dos 0 para denotar que no hay flujos externos en ese nodo). El caso de flujo externo fijo se corresponde con el caso $l_i^e = u_i^e$.

En contra de lo que pueda parecer, las redes con flujos externos no son más generales, porque siempre se puede transformar una red en la que pueda entrar o salir flujo en otra equivalente de la que no puede entrar o salir flujo. Esto nos permite prestar atención únicamente a los parámetros definidos sobres los arcos y olvidarnos de los parámetros definidos sobre los nodos. Para ello se define un nuevo nodo. Si el flujo externo de un nodo i es positivo, da lugar a un arco que va desde el nuevo nodo al nodo i. Si el flujo externo del nodo i es negativo, da lugar a un arco que va desde el nodo i al nuevo nodo.

3.2. Formulación.

El problema general de flujo en redes a coste mínimo lo definiremos en principio sobre redes sin flujos externos. Una vez que se han eliminado los flujos externos de una red, es natural imponer la restricción conocida como *conservación de flujo*, que establece que el flujo que entra en un nodo ha de ser igual al flujo que sale (ningún

nodo puede generar o eliminar flujo, ya que esto sería como tener un flujo externo).

Definición 3.1. Dada una red con flujo, el problema de flujo en redes con coste mínimo consiste en determinar el flujo que ha de pasar por cada arco de tal manera que las restricciones impuestas por las capacidades se cumplan y que el coste sea mínimo. Los flujos son las variables de decisión del problema de optimización: elegir los flujos, cumpliendo las restricciones impuestas, que minimicen el coste total del flujo que pasa por la red.

A continuación mostramos que todo PFCM se puede ver como un problema de programación lineal. Supongamos que el grafo subyacente a la red es $G = (N, M)$ y denotemos por M_{O_i} el conjunto de arcos que se originan en el nodo i y por M_{T_i} el conjunto de arcos que terminan en el nodo i. Entonces, basta formular el problema del siguiente modo:

$$\begin{aligned} \text{minimizar} \quad & \sum_{k \in M} c_k \times f_k \\ \text{sujeto a:} \quad (1) \quad & \sum_{k \in M_{O_i}} f_k - \sum_{k \in M_{T_i}} f_k = 0 \quad i \in N \\ (2) \quad & f_k \leq u_k \qquad k \in M \\ (3) \quad & f_k \geq l_k \qquad k \in M \end{aligned}$$

La función objetivo es el coste que supone el flujo de los f_k . Por otro lado, las restricciones en (1) no son más que las restricciones de conservación de flujo, una por cada nodo. Además, por cada arco tenemos dos restricciones de capacidad, una para las cotas superiores (2) y otra para las cotas inferiores (3).

Aunque todos los elementos de la expresión parecen lineales, estaría bien tener una representación matricial del problema que nos permita ver más claramente la matriz de restricciones y el vector con las constantes del lado derecho. Recordemos que $\bar{B}$ denota la matriz de incidencia del grafo G. Dado un vector v supondremos que es un vector fila y denotaremos su traspuesto por $v^{'}$. Entonces, si denotamos por c, f, u y l los vectores de costes, flujos, cotas superiores y cotas inferiores, es fácil ver que el anterior problema se puede expresar como:

$$\begin{aligned} \text{minimizar} \quad & c^{'} \times f \\ \text{sujeto a:} \quad & (1) \quad \bar{B}_{n\times m} \times f = 0 \\ & (2) \quad \bar{I}_{m\times m} \times f \leq u \\ & (3) \quad \bar{I}_{m\times m} \times f \geq l \end{aligned}$$

Observando que todo PFCM es un problema de programación lineal. Por tanto, podríamos aplicar directamente el método Simplex para estudiar este problema u otros algoritmos específicos que explotan la estructura especial de este problema para conseguir una mayor velocidad de resolución.

Aunque hemos comentado que no hay pérdida de generalidad en trabajar sin flujos externos y que el problema general de flujo es más fácil de presentar, hay un caso de flujos externos que es muy fácil de incorporar en el modelo, y es el caso en el que, para todo nodo i, $c_i^e = 0$ y $l_i^e = u_i^e = e_i$ es decir, los flujos externos son fijos y tienen coste 0. En este caso, los e_i entran en el problema de optimización a través de las ecuaciones de conservación de flujo:

$$\begin{aligned} \text{minimizar} \quad & \sum_{k\in M} c_k \times f_k \\ \text{sujeto a:} \quad & (1) \quad \sum_{k\in M_{O_i}} f_k - \sum_{k\in M_{T_i}} f_k = e_i \quad i \in N \\ & (2) \quad f_k \leq u_k \qquad k \in M \\ & (3) \quad f_k \geq l_k \qquad k \in M \end{aligned}$$

Con lo que las restricciones de conservación de flujo pasan a decir que el flujo que entra en un nodo menos el que sale debe de ser igual al flujo externo.

En muchas aplicaciones en la vida real, los flujos representarán personas, vehículos o simplemente objetos indivisibles. En este caso, habrá que restringirse al caso en el que los flujos tienen que tomar valores enteros y, entonces, tenemos un problema de programación lineal entera, que sabemos que, en general, no es un problema fácil; el

Simplex ya no sirve y no tiene por qué haber algoritmos polinomiales. Sin embargo, la estructura de los PFCM permite bordear este problema de modo satisfactorio. Para ello es clave el siguiente resultado, consecuencia de la propiedad de unimodularidad, Papadimitriou y Steiglitz, (1982).

Proposición 3.1. Si todas las capacidades de un PFCM toman valores enteros, entonces existe un óptimo en el que todos los valores son enteros. De hecho, el siguiente resultado, que es todavía más fuerte también es cierto.

Proposición 3.2. Si todas las capacidades de un PFCM toman valores enteros, entonces, en toda solución básica factible, todos los valores son enteros.

Estos resultados son de suma importancia para poder resolver problemas de programación. Hoy en día hay una gran cantidad de problemas enteros y/o combinatorios para los que se ha conseguido probar que son fáciles. En casi todos los casos, tenemos que detrás de los algoritmos correspondientes se esconde el hecho de que existe un óptimo con valores enteros. La clase de los PFCM es una de las clases más grandes en las que se cumple esta propiedad.

En nuestro caso estas proposiciones son básicas para poder considerar nuestro problema como un PFCM, ya que si los valores en el óptimo no fuesen enteros, querría decir que una actividad sería dada por más de un monitor, cuestión inaceptable en nuestro caso.

Capítulo 4

Adaptación del problema a un PFCM.

En este caso se ha optado por adaptar el problema a la formulación de un PFCM con flujos externos fijos, con las siguientes peculiaridades:

- Las unidades de flujo que circularan por la red serán horas.
- En nuestro modelado los costes en los arcos representarán las preferencias de los monitores, con lo que el objetivo pasa a ser minimizar preferencias. Dando los valores más bajos a las actividades y horas preferidas por los monitores, devolverá el resultado que, dentro de lo factible, respeta al máximo las preferencias, consiguiendo la mayor "felicidad" posible en el conjunto de todos los monitores.

Con lo que el problema se podría formular de la siguiente forma, si consideramos que en cada arco el coste c_k lo cambiamos por dichas preferencias que llamaremos p_k:

$$\begin{aligned} \text{minimizar} \quad & \sum_{k \in M} p_k \times f_k \\ \text{sujeto a:} \quad (1) \quad & \sum_{k \in M_{O_i}} f_k - \sum_{k \in M_{T_i}} f_k = e_i \quad i \in N \\ (2) \quad & f_k \leq u_k \qquad k \in M \\ (3) \quad & f_k \geq l_k \qquad k \in M \end{aligned}$$

4.1. Diseño del grafo.

Para poder adaptar adecuadamente el problema en un PFCM una de las claves es el diseño del grafo. Por su propia estructura y propiedades permitirá controlar el mayor número de condiciones exigidas al problema, en la Figura 4.1 se muestra el grafo empleado para la resolución del problema. La zona inferior indica el número de nodos y arcos.

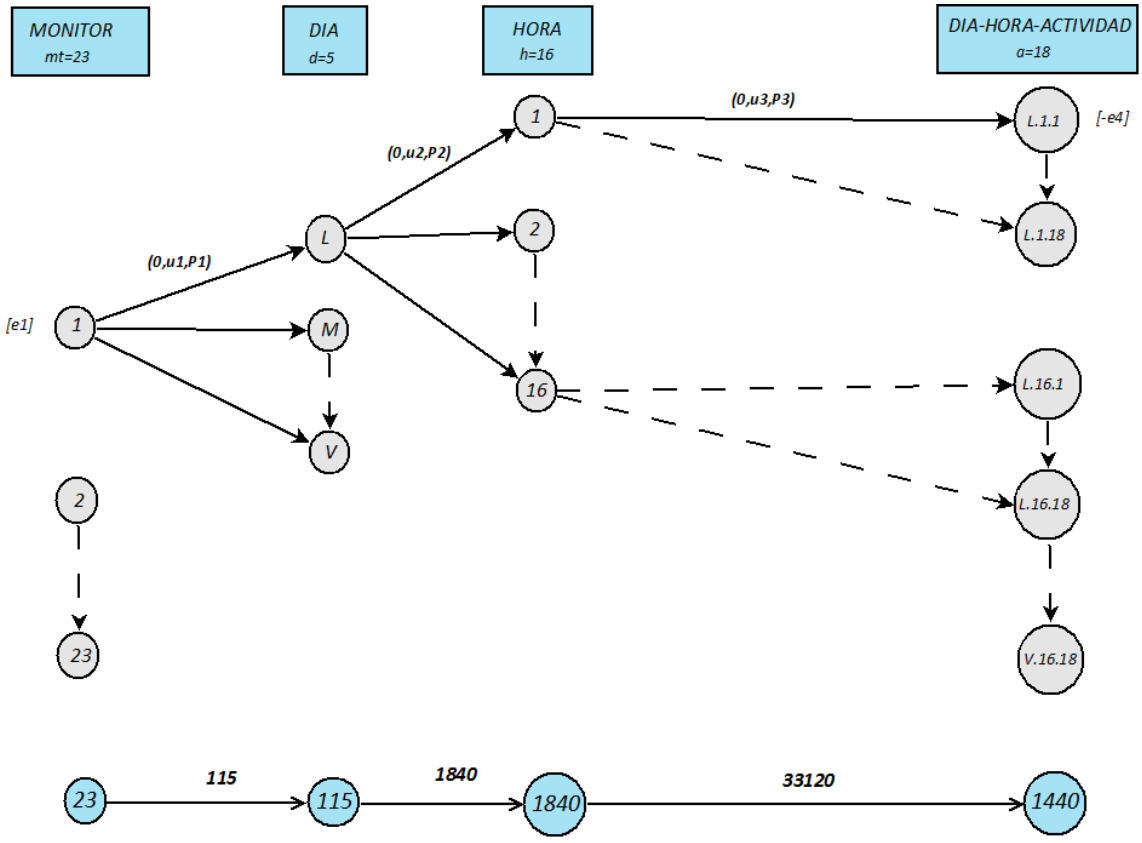

Fig. 4.1: Grafo asociado al problema.

Como hemos comentado anteriormente el diseño del grafo es muy importante para controlar ciertos aspectos del problema. Si hubiésemos optado por el diseño clásico de un problema de transporte sólo con nodos inicio y final, es decir, en este caso *"monitores"* y *"Día-hora-actividad"*, no podríamos delimitar el máximo de horas diarias, ni que cada monitor impartiese una actividad a la hora.

Para poder controlar el máximo de horas que un monitor puede impartir en un día ($u1$) se introdujo el grupo de nodos *"Día"*, y para que cada monitor solo pueda impartir una actividad por hora ($u2 = 0$ o 1) se introdujo el grupo de nodos *"horas"*. Aunque este diseño aumente la complejidad del grafo, resulta una opción muy eficiente para controlar dichos aspectos ya que si no tendríamos que incluir variables binarias en el problema.

4.2. Variables y restricciones del problema.

Las variables de decisión del problema son los flujos f_k que circularan por el grafo asociado al problema (véase Figura 4.1), en este caso horas, donde podemos observar que al tratarse de:

$m_t = 23$; Monitores de la instalación.
$d = 5$; Días de trabajo semanal.
$h = 16$; Horas de apertura diarias de la instalación.
$a = 18$; Actividades diferentes a impartir.

Se obtiene un total de $m_t \times d \times h \times a = 33{,}120$ variables que tomaran valor 0 o 1, debido al diseño del grafo y a las propiedades del PFCM, donde un 1 representará que el monitor m_t, el día d, a la hora h impartirá la actividad a, y un 0 que no la impartirá

El grafo asociado al problema consta de $n = 3{,}418$ nodos y $m = 35{,}075$ arcos, por tanto tendremos 3,418 restricciones de conservación de flujo y 70,150 restricciones de capacidad, haciendo un total de 73,568 restricciones.

Nótese que estamos enfrentándonos a un problema con 33.120 variables y 73.568 restricciones, números nada despreciables, dando lugar a un trabajo laborioso de programación, ya que se trabajará con vectores de dichas dimensiones y matrices de $n \times m = 119.886.350$, es decir, prácticamente matrices de 120 millones de datos.

Con este número tan elevado de variables se hace muy complicado tratar el problema como programación entera ya que el tiempo de ejecución sería muy elevado, pero al considerarlo como un PFCM se puede considerar un problema "fácil".

Otro dato a tener en cuenta es que la ejecución del programa realizada en un ordenador portátil de 4 Gb de memoria RAM emplea el 90 % de su capacidad de memoria en el momento de cargar los datos, lo que nos hace comprender las dimensiones del problema que estamos tratando.

4.3. Definición de los parámetros del grafo.

En este momento nos disponemos a definir los parámetros asociados a los arcos y a los nodos en nuestro modelo, para adecuarlos a las condiciones especiales del problema que nos ocupa.

Parámetros asociados a los arcos:

- **Vectores de límites inferiores de los arcos (l).**

  En este caso se consideran todos igual a cero.

- **Vectores de límites superiores de los arcos (u).**

  Este valor limita la posibilidad de impartir una actividad, hora o día en concreto para cada monitor, por tanto $u = 0$ implica imposibilidad. Al contrario que las preferencias, la introducción de límites superiores de flujo en cada arco sí que reducirá la región factible por lo que se intentara introducir el menor número posible de ceros, reservándolos para los casos en los que no quede otra opción.

  $u1$; días posibles por monitor (si no es posible el día, $u1 = 0$).

  $u2$; horas posibles diarias por monitor.

  $u3$; actividades posibles por monitor.

- **Vectores de preferencias (P):**

  $P1$; preferencias diarias por monitor.

  $P2$; preferencias por horas.

  $P3$; preferencia de actividades.

  Con la introducción de estas preferencias se podría estudiar la posibilidad de eliminar algún monitor en función de la demanda, o días de monitores, también se podrá adaptar los horarios de los monitores a sus preferencias tanto de horas

como de actividades a impartir. Al tratarse de preferencias estas no acotan la región factible, con lo que el conjunto de soluciones posibles no se ve reducida.

Nótese que el primer objetivo del programa es obtener una solución factible que cubra la demanda diaria de actividades en la instalación, pasando a un segundo plano la "felicidad" de los monitores.

Estas preferencias serán asignadas por el coordinador y/o los monitores, según los criterios de la dirección del centro.

Parámetros asociados a los nodos:

- **Vectores de flujos externos.**

  $e1$; indicará el número de horas que puede impartir cada monitor por semana (oferta).

  $e4$; indicará que actividades se deberán impartir en el gimnasio cada hora en los distintos días. Estos datos de entrada corresponden al horario de actividades existentes (demanda a cubrir) y serán proporcionados por la dirección.

  Los vectores de flujos externos de los nodos intermedios serán todos iguales a cero.

4.4. Problemas de adaptación a un PFCM.

Como vimos en el Capítulo 2, la falta de flexibilidad en el número de horas de *"sala"* y *"actividades"* que debe impartir cada monitor a la semana es un inconveniente para adecuar el problema a un PFCM, ya que en el grafo asociado al problema cada nodo asignado a los monitores debe tener un flujo externo entrante único. Para resolver este problema se propone ejecutar el programa por pasos, es decir primero se asignarán unas de las horas, por ejemplo las de sala, y después de eliminar las horas asignadas en el primer paso se asignarán las restantes, con lo que en cada paso el problema si que será un PFCM.

Como veremos en el siguiente Capítulo el programa se diseñará para adecuarse a distintas condiciones, una de ellas será contemplar que no existe esta restricción de rigidez en el número de horas de *"sala"* y *"actividades"*, considerándolas por igual, con lo que la adaptación a un PFCM será más sencilla y aumentará la flexibilidad de la

instalación.

Para ilustrar el problema que surge con estas restricciones de rigidez vamos a suponer un problema muy sencillo con 2 monitores, 1 día, 4 horas y 2 actividades, considerando que el monitor 1 prefiere las primeras horas y la actividad 1, y el monitor 2 las últimas horas y la actividad 2.

Por tanto, tenemos $m_t \times d \times h \times a = 16$ variables. El grafo asociado al problema tiene $n = 20$ nodos y $m = 26$ arcos (véase 4.2) es decir, tenemos 20 restricciones de conservación de flujo y 52 restricciones de capacidad.

En la Figura 4.3 se muestra la solución obtenida en la ejecución en un paso, devolviendo un valor objetivo de 18.

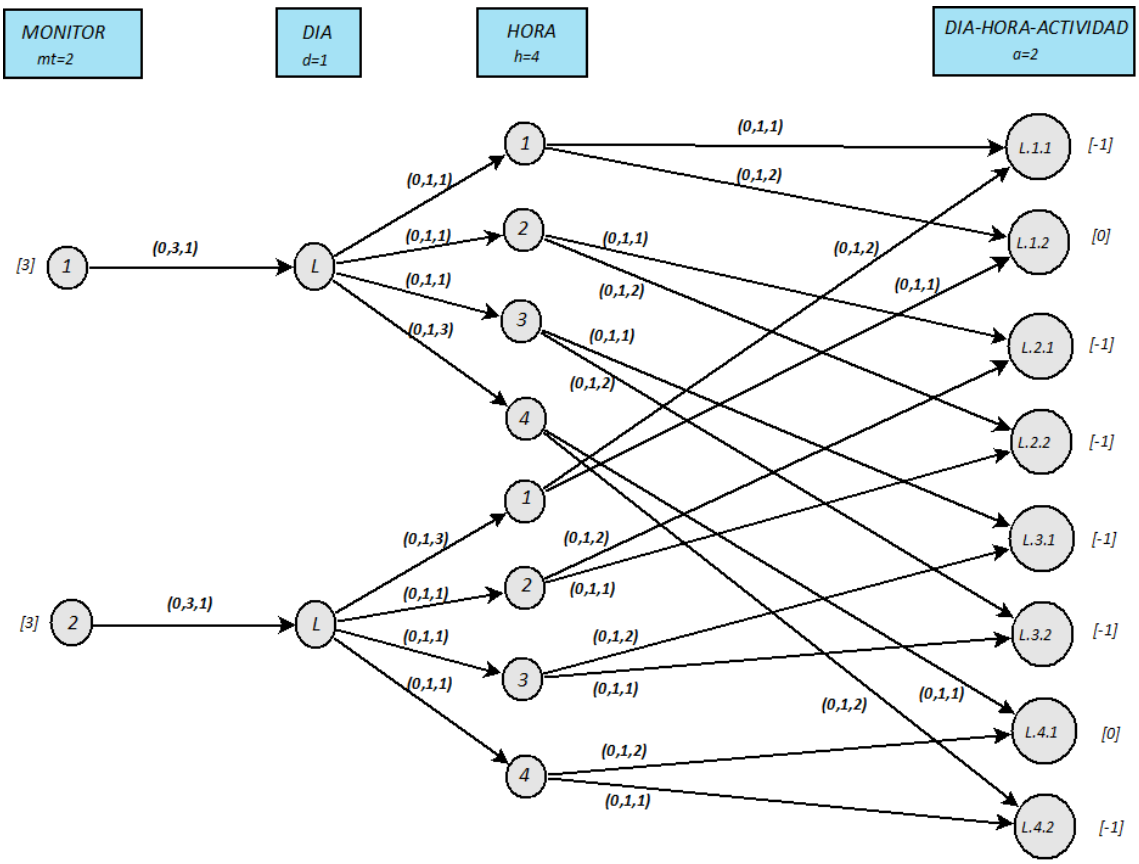

Fig. 4.2: Grafo asociado al ejemplo.

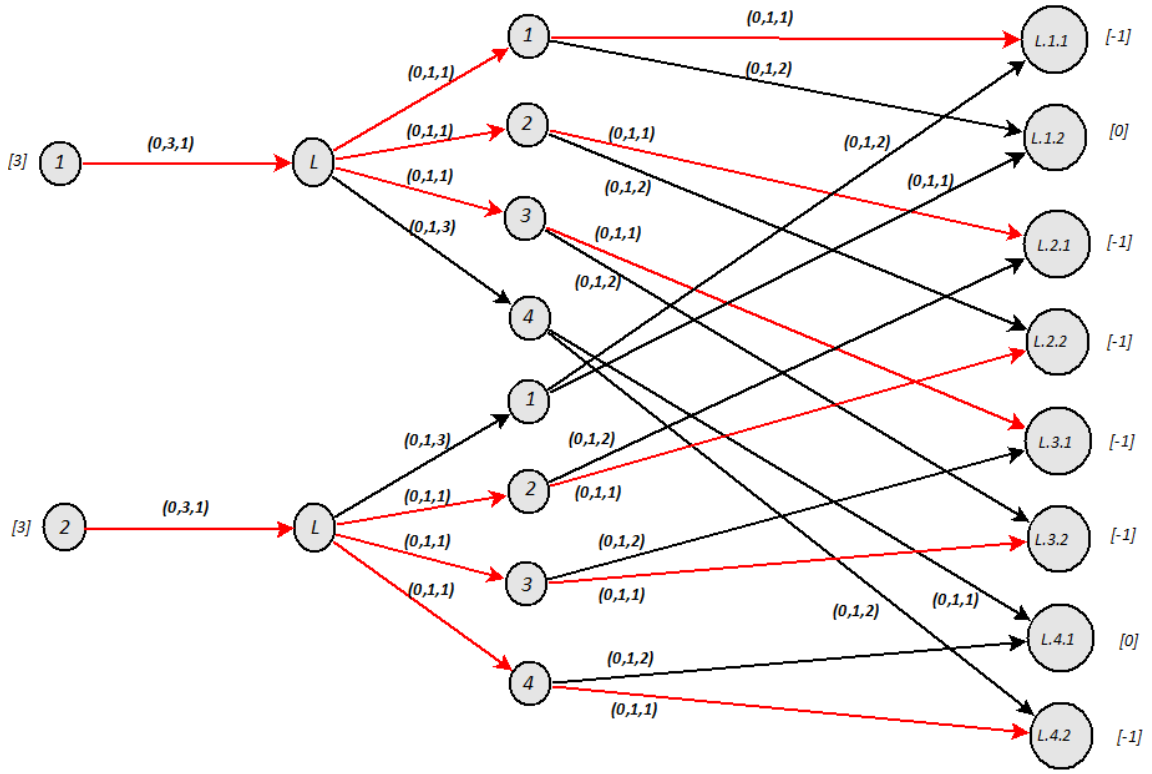

Fig. 4.3: Solución del ejemplo en un paso.

Pasemos ahora a resolver el ejemplo en dos pasos asignando primero la actividad 1, con la restricción de que el monitor 1 debe impartir una actividad 1 y dos el monitor 2. En la figura 4.4 se muestra el grafo asociado y una de las posibles soluciones óptimas en el primer paso, obteniendo un valor objetivo de 11.

Para ejecutar el segundo paso es necesario eliminar las horas ya asignadas en el primer paso, obteniendo el grafo y la solución que se muestra en la Figura 4.5, con un valor objetivo de 11. Por tanto la solución total en dos pasos (véase Figura 4.6) es 22, ligeramente superior al valor obtenido en un paso, esto nos demuestra que bajo ciertas condiciones la ejecución en dos pasos no asegura la obtención del óptimo.

Nótese que la diferencia en el valor objetivo obtenida por ambos métodos es debida a que la ejecución en dos pasos es más restrictiva, ya que obligamos a los monitores a impartir un número de actividades concretas. Si metiésemos estas mismas restricciones exteriormente al problema en un paso los objetivos coincidirían.

Imaginemos ahora que los monitores no tienen preferencias de actividad, tomando

$P3 = 1$. En este caso es fácil comprobar que con la resolución en dos pasos se obtiene el mismo valor objetivo que con la resolución en un solo paso, esto nos demuestra que a veces la resolución en dos pasos sí nos devolverá el óptimo.

También es cierto que, en un caso extremo en el que la flexibilidad de los monitores esté muy reducida, es decir, que tengan pocas actividades que puedan impartir o una limitación horaria, podría darse el caso que con la resolución en dos pasos no obtengamos una solución factible.

En todas las pruebas realizadas en el caso real que nos ocupa nunca ocurrió, pero podría ocurrir, por lo que se sigue trabajando en desarrollar un modelo que nos permita modelar correctamente el problema cuando existen este tipo de restricciones de rigidez. En el Capítulo 7 se discutirán algunas de las líneas de trabajo que se contemplan para resolverlo.

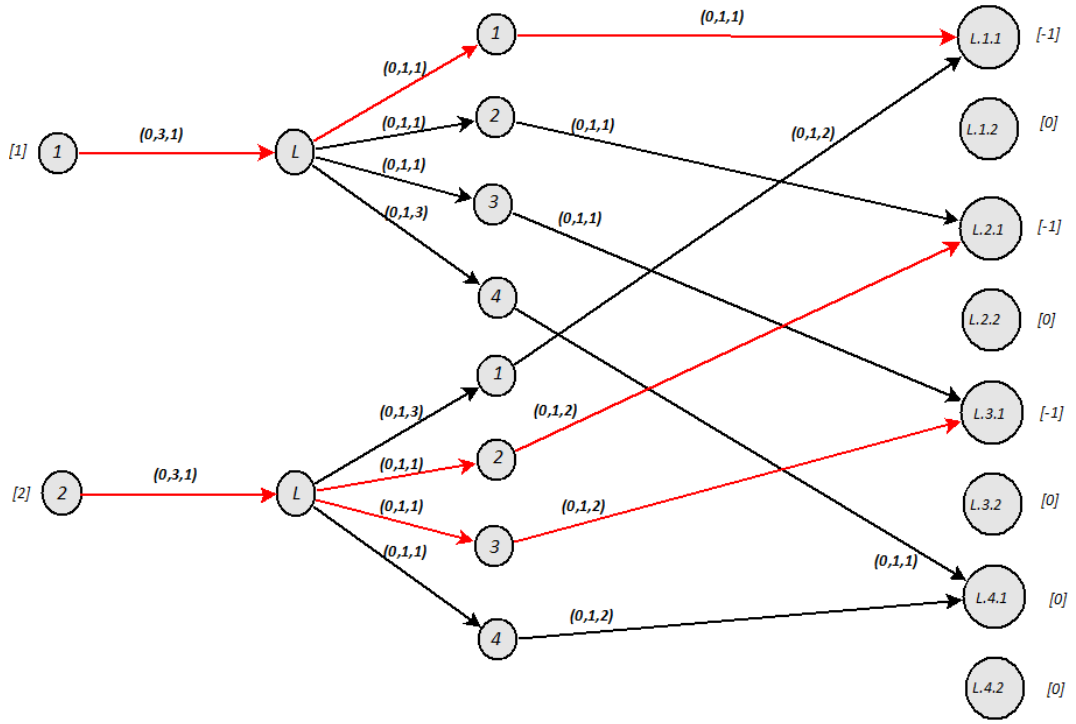

Fig. 4.4: Solución al ejemplo (paso 1).

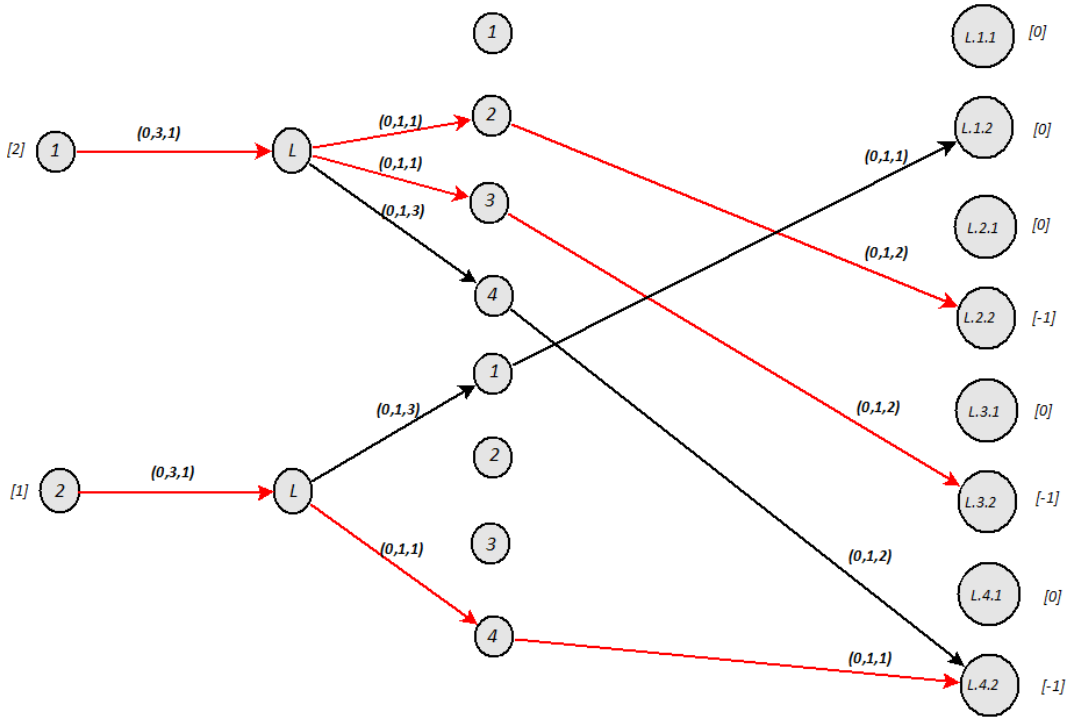

Fig. 4.5: Solución al ejemplo (paso 2).

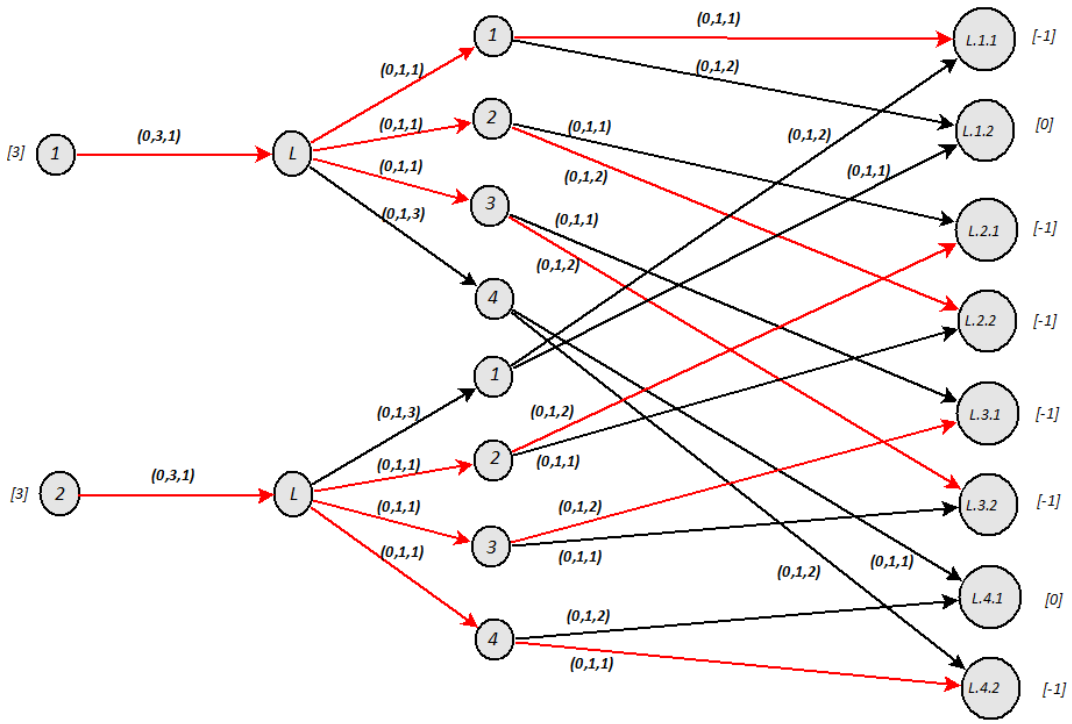

Fig. 4.6: Solución del ejemplo en dos pasos.

Capítulo 5

Características del Programa.

5.1. Esquema del programa.

En la Figura 5.1 se muestra el esquema general del programa, cabe destacar que las preferencias introducidas nunca afectarán a la región factible, sino que se emplean para obtener la solución óptima de forma que maximicen las preferencias conjuntas según el criterio elegido.

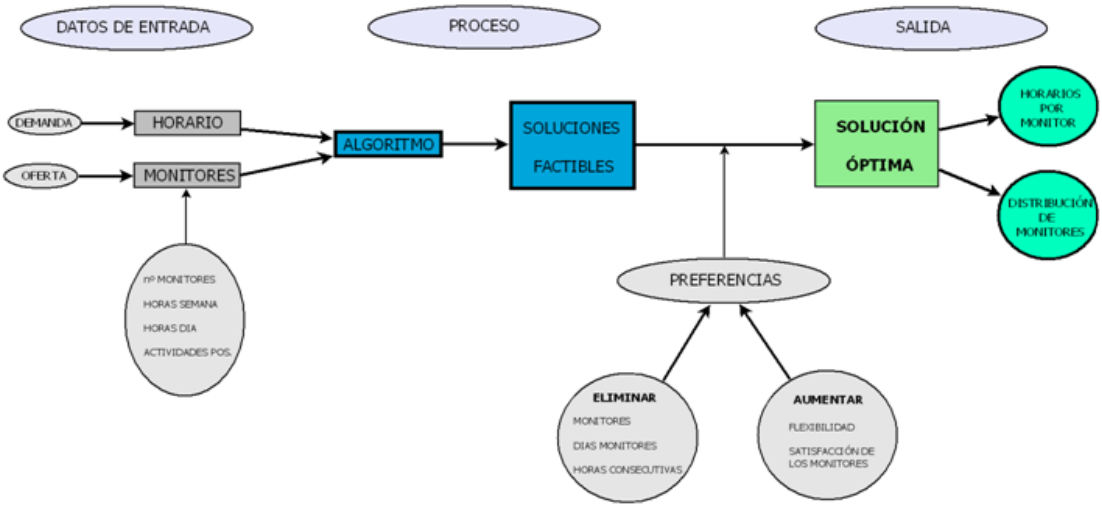

Fig. 5.1: Esquema general del programa.

5.2. Opciones de resolución.

Para aumentar la flexibilidad del programa y que pueda ser empleado en diversas situaciones y bajo criterios dispares, se considera oportuna la inclusión de diferentes parámetros que permitirán ejecutar el programa de varias formas: en función del tipo de resolución, del horario a cubrir y de los criterios de preferencia.

Resolución.

En función del tipo de resolución se consideraron 3 formas de resolver el problema. Para ello se introdujo el parámetro *"Res"*, a elegir por el usuario:

$Res = 1$; Primero se asignan las horas de sala y a continuación las de actividades.
$Res = 2$: Primero se asignan las horas de actividades y después las de sala.
$Res = 3$; Se asignan todas las horas a la vez sin distinguir el tipo de hora, este caso estaría reservado para instalaciones en las que no se diferencie el tipo de hora a impartir por los monitores, considerando solo el número de horas que debe dar a la semana.

Este último caso parece el más razonable en este tipo de instalaciones, ya que daría mayor flexibilidad para adaptarse a los cambios en la demanda, ya que esta sufre múltiples variaciones a lo largo del año: dependiendo de la temporada y de los festivos.

Tipo de horario.

La instalación posee dos tipos de horarios claramente diferenciados, un horario normal durante el año y otro vacacional. El vacacional comprende las navidades y los meses de verano, donde el horario sufre cambios considerables. Para adaptar el programa a estos dos tipos de horario, se ha introducido el parámetro "T'' que en función de su valor, el programa considera o no las actividades marcadas como 3ª preferencia (véase Figura 5.4).

$T = 1$; considera solo la 1ª y 2ª preferencia de actividades.
$T = 3$; considera todas las actividades posibles de los monitores.

Es decir con $T = 3$ el programa se adecua al horario vacacional, dando la posibilidad a los monitores a impartir un mayor número de actividades que el resto del año, lo que aumenta considerablemente la flexibilidad del centro y permite cubrir todo el horario con un número menor de monitores.

Criterio.

Con el parámetro *"Cri"* se contempla la posibilidad de ejecutar el programa de 2 formas: dando prioridad a las preferencias horarias o a las actividades elegidas por los monitores:

$Cri = 1$; Prioridad a las actividades.
$Cri = 2$, Prioridad a las horas.

Como vimos en el Capítulo 2 la variabilidad de horas a impartir por los monitores nos hace plantearnos la posibilidad de crear dos grupos de monitores diferenciados, que llamaremos *"monitores A"* y *"monitores B"*

Monitores A: Tienen una mayor carga de trabajo y además los únicos que pueden cubrir las horas de sala (monitores 1-10).
Monitores B: Tienen menor carga de trabajo y no pueden cubrir las horas de sala.

Parece razonable considerar que los dos grupos de monitores sean tratados de forma diferente en cuanto a la consideración de sus preferencias, dando cierta prioridad a los monitores que más horas deben impartir (criterio elegido por la dirección).

Para tener en cuenta dicha distinción se introdujeron ciertas constantes en el programa, que en función del criterio elegido para la asignación modifica los valores de preferencia introducidos por los monitores, dando preferencia a los *monitores A* frente a los *B*.

Por ejemplo si elegimos el *criterio 1*, teniendo en cuenta la distinción entre monitores, el orden de preferencia de asignación considerado es el siguiente:

1º Monitores A, actividad preferida, hora preferida.

2º Monitores A, actividad preferida, hora no preferida.

3º Monitores B, actividad preferida, hora preferida.

4º Monitores B, actividad preferida, hora no preferida.

5º Monitores A, actividad no preferida, hora preferida.

6º Monitores A, actividad no preferida, hora no preferida.

7º Monitores B, actividad no preferida, hora preferida.

8º Monitores B, actividad no preferida, hora no preferida.

No obstante, este criterio podría variarse en función de las preferencias de elección de la dirección del centro.

5.3. Preferencias de la dirección y monitores.

Uno de los trabajos más laboriosos en el desarrollo del programa fue determinar las distintas preferencias y criterios a tener en cuenta tanto por la dirección como por los monitores. Para ello se realizaron innumerables reuniones con el coordinador del centro, responsable de los horarios, ya que sus criterios serían los más importantes a tener en cuenta a la hora de adaptar el programa a la instalación.

Las decisiones más importantes tomadas en estas reuniones fueron:

- Tomar los valores 1, 2 y 3 para las preferencias de actividades ($P3$), dejando el valor 3 sólo para el tipo de horario vacacional ($T = 3$).
- Tomar $u1 > 0$ y $u2 = 1$, es decir, permitir a todos los monitores trabajar todos los días y a todas horas. Esta decisión parece muy razonable ya que de esta forma no se reduce la región factible, si se buscase eliminar algún día de trabajo de cierto monitor, por petición propia, sería más conveniente variar las preferencias de días ($P1$) que no afectaría a la región factible. En este caso se tomó $P1 = 1$ para todos los monitores.

  Este parámetro también podría emplearse para dar cierta prioridad general a algún monitor ya sea por antigüedad o por algún otro criterio a considerar.
- Determinar el orden de prioridades entre horas y actividades que se detalló en la sección 5.2 para los dos tipos de criterios de resolución contemplados.
- Decidir que un monitor no podría impartir más de 2 horas seguidas de cierta actividad, en este caso “cycling”, considerada de alto gasto energético. Para resolver este problema sin incluir variables binarias se consideró realizar un

postprocesado después de obtener la solución, analizando si existen 3 actividades de este tipo seguidas, si es así, el programa cambia la preferencia de esa tercera hora a un valor elevado y se vuelve a ejecutar el programa hasta que deje de ocurrir.

Todas las decisiones tomadas anteriormente adaptan el programa a las condiciones exigidas por esta instalación, pero podrían adecuarse a las exigencias de otros centros con diferentes problemáticas.

Es cierto que, todos los parámetros incluidos en el programa parecen complicarlo un poco, pero son necesarios para dotarlo de flexibilidad y así poder adaptarse a múltiples situaciones.

Por otro lado, para conocer las prioridades e inquietudes de los monitores se realizó una encuesta personal a los monitores con mayor carga de trabajo (monitores A), donde se conoció:

- Su grado de satisfacción con el horario actual.
- Preferencias de actividades a impartir.
- Preferencia horaria.
- Horas de entrenamientos personales y su importancia.
- Inquietudes personales.
- Grado de satisfacción con el horario de vacaciones.
- Mejoras a realizar en el horario actual.
- Etc.

En la Figura 5.2 se muestra el resultado a alguna de estas preguntas, pero sin duda el resultado más importante de esta encuesta fue conocer de primera mano la opinión de cada uno de ellos siendo de gran ayuda para poder adaptar los distintos parámetros del programa y así obtener una solución capaz de satisfacer a todos.

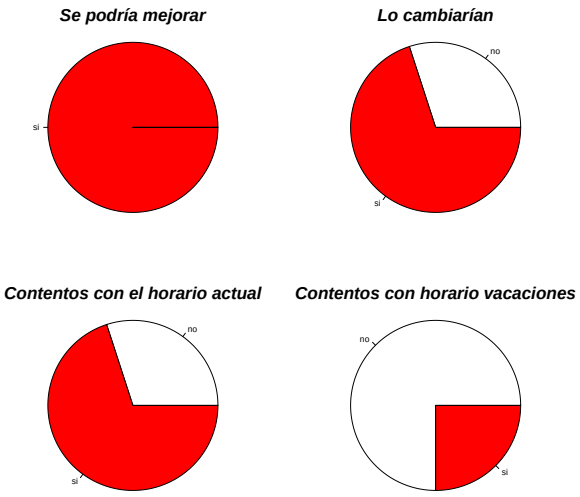

Fig. 5.2: Opinión de los monitores.

5.4. Datos de entrada.

Para el proceso de introducción de datos se optó por crear una hoja *Excel "Horarios.xlsx"*, por su fácil manejo y conocimiento para la mayoría de los posibles usuarios. Diseñada para que en ella se introduzcan todos los datos y parámetros necesarios para la ejecución del programa, creando diferentes pestañas para su correcta ordenación. Dicha hoja está conectada con *R* con la ayuda de la *librería xlsx*, posibilitando la lectura de datos de forma directa incluso cuando la hoja está abierta.

A continuación se detallan los pasos a seguir para la correcta introducción de los datos.

5.4.1. Horario a cubrir (demanda).

En la pestaña *"Horario de entrada"* se introducirá el horario semanal de actividades a cubrir de la instalación, así como el número de monitores necesarios en sala y piscina por hora. De donde se obtendrán los vectores de flujos externos $e4$. En la Figura 5.3 podemos observar el horario de actividades del lunes donde un valor entero en las columnas de sala y piscina indica el número de monitores necesarios, y en las columnas de actividades, un 1 indica que hay una hora de actividad y un 0 que no la hay.

DEMANDA e4		SALA	PISCI	CYC.	MET	CROSS	PILA	TAI	YOGA	ZUMBA	NORD	SWIM.	RUNING	AQ. G.	AQ. A.	AEROSTL.	FIT.	A. STEP	CARD. C.
LUNES	7	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	2	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	11	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	12	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	13	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	14	2	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	15	2	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	16	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	17	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	18	2	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	19	3	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
	20	3	1	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0
	21	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	22	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 5.3: Horario a cubrir el lunes (demanda).

5.4.2. Datos de los monitores (oferta).

Actividades posibles de los monitores.

En la Figura 5.4 se pueden observar las distintas actividades que puede impartir cada monitor (pestaña *"limitaciones"*), un 1 indica las actividades preferentes, con un 2 la segunda elección y con un 3 las actividades que podrán impartir a mayores cuando el programa se ejecute en modo vacacional ($T = 3$). De esta tabla se obtendrán posteriormente los vectores $u3$ y $P3$.

En este caso sólo se tomaron tres valores diferentes, asignando a casi todas las actividades preferidas un 1, por decisión del coordinador. A nuestro entender sería interesante considerar la posibilidad de introducir un mayor número de valores posibles, permitiendo a los monitores que claramente mostrasen sus preferencias de actividades y así obtener una solución más satisfactoria para ellos.

Horas a impartir y horas máximas diarias.

En estas tablas (véase Figura 5.5) se introducirán: en las dos primeras columnas de la tabla de la izquierda las horas de sala y actividades que deberá impartir cada monitor semanalmente por contrato (obtendremos $e1$), y en el resto de columnas el número máximo de horas de sala diarias ($u1$). En la tabla de la derecha las horas máximas de actividades a impartir diariamente por cada monitor ($u1$ actividades).Nótese que se diferencian los vectores $e1$ y $u1$ entre horas de sala y actividades para las resoluciones 1 y 2, cuando la resolución sea de tipo 3 el programa las sumará.

		ACTIVIDADES POR MONITOR INTRODUCIR VALORES 1, 2 y 3 (1 opción preferente)(3 posibilidad vacaciones)																	
		SALA	PISCI	CICLIN	MET	CROSS	PILA	TAI	YOGA	ZUMBA	NORD	SWIM.	RUNING	AQ. G.	AQ. A.	A. ST.	FIT	A. S	CRD.. C.
MONITOR	1	1	1	1	1	3	3	0	0	1	0	0	0	1	1	1	1	1	3
	2	1	1	1	1	3	0	0	0	1	0	0	0	1	1	1	1	1	3
	3	1	1	0	0	3	0	0	0	0	0	3	1	0	0	0	0	0	0
	4	1	1	0	0	0	0	0	0	0	0	1	0	3	0	0	0	0	0
	5	1	1	1	1	3	1	0	0	0	0	0	0	2	1	3	3	1	2
	6	1	1	1	1	3	0	0	0	0	0	3	0	1	0	0	0	0	0
	7	1	1	1	0	3	0	0	0	0	3	3	3	3	0	0	0	0	0
	8	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9	1	1	1	0	3	0	0	0	0	0	3	3	0	0	0	0	0	0
	10	1	1	1	1	1	1	0	3	0	0	0	0	3	0	0	1	3	0
	11	0	0	0	3	0	0	0	0	0	0	0	0	1	3	0	0	0	0
	12	0	0	1	0	3	0	0	0	0	3	3	3	3	0	0	0	0	0
	13	0	0	1	3	1	3	0	0	1	1	0	0	3	0	0	3	0	0
	14	0	0	1	0	1	0	0	0	0	0	3	3	0	0	0	0	0	0
	15	0	0	1	3	3	0	0	0	0	3	3	0	1	3	0	0	0	0
	16	0	0	3	1	3	0	0	0	0	0	0	0	3	0	0	3	3	3
	17	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	18	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	19	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	20	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0
	21	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	22	0	0	3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	23	0	0	1	1	0	0	0	0	0	3	0	0	1	1	1	1	1	1

Fig. 5.4: Actividades posibles de los monitores.

		HORAS MÁXIMAS DIA SALA-PISCINA							HORAS MÁXIMAS DIA ACTIVIDADES				
		e1		U1					U1 Actividades				
		SALA	ACT	L	M	M	J	V	L	M	M	J	V
MONITOR	1	20	10	5	5	5	5	5	3	3	3	3	3
	2	20	10	5	5	5	5	5	3	3	3	3	3
	3	20	3	4	4	4	4	4	2	2	2	2	2
	4	20	2	4	4	4	4	4	4	4	4	4	4
	5	15	7	5	5	5	5	5	4	4	4	4	4
	6	20	3	5	5	5	5	5	3	3	3	3	3
	7	20	2	5	5	5	5	5	2	2	2	2	2
	8	30	0	7	7	7	7	7	0	0	0	0	0
	9	16	3	5	5	5	5	5	4	4	4	4	4
	10	15	8	6	6	6	6	6	4	4	4	4	4
	11	0	1	6	6	6	6	6	4	4	4	4	4
	12	0	3	6	6	6	6	6	2	2	2	2	2
	13	0	5	6	6	6	6	6	3	3	3	3	3
	14	0	2	6	6	6	6	6	4	4	4	4	4
	15	0	2	6	6	6	6	6	4	4	4	4	4
	16	0	1	6	6	6	6	6	4	4	4	4	4
	17	0	4	6	6	6	6	6	4	4	4	4	4
	18	0	3	6	6	6	6	6	4	4	4	4	4
	19	0	3	6	6	6	6	6	4	4	4	4	4
	20	0	4	6	6	6	6	6	4	4	4	4	4
	21	0	0	6	6	6	6	6	4	4	4	4	4
	22	0	1	6	6	6	6	6	4	4	4	4	4
	23	0	7	6	6	6	6	6	4	4	4	4	4

Fig. 5.5: Horas a impartir y horas máximas diarias de los monitores.

Preferencias horarias.

En la pestaña *"Prioridades"* se introducirán las preferencias horarias de cada mo-

nitor para cada día de la semana (con las que obtendremos P_2), dando valores de 0 a 6 donde 0 es la máxima preferencia, el valor 6 se reserva para marcar las horas de entrenamiento personal *"inamovibles"* de cada uno de los monitores. En la Figura 5.6 se muestran las preferencias de 5 de los monitores, donde podemos comprobar la disparidad de sus preferencias.

		INTRODUCIR VALORES DE 0 A 5 (0 MAXIMA PRIORIDAD) (6 ENTRNAMIENTOS PERS. FIJOS)															
		PREFERENCIAS HORARIAS															
MONITOR	(P2)	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	L	0	0	0	0	2	5	5	5	5	5	5	1	1	5	5	5
	M	0	0	0	0	2	5	5	5	5	5	5	1	1	5	5	5
	M	0	0	0	0	2	5	5	5	5	5	5	1	1	5	5	5
	J	0	0	0	0	2	5	5	5	5	5	5	1	1	5	5	5
	V	0	0	0	0	2	5	5	5	5	5	5	1	1	5	5	5
2	L	0	0	0	0	2	5	5	5	5	5	5	1	1	5	5	5
	M	0	0	0	0	2	5	5	5	5	5	6	1	1	5	5	5
	M	0	0	0	0	2	5	5	5	5	5	5	1	1	5	5	5
	J	0	0	0	0	2	5	5	5	5	5	5	1	1	5	5	5
	V	0	0	0	0	2	5	5	5	5	5	5	1	1	5	5	5
3	L	5	5	5	0	0	0	0	5	5	5	5	5	5	5	5	5
	M	5	5	5	0	0	0	0	5	5	5	5	5	5	5	5	5
	M	5	5	5	0	0	0	0	5	5	5	5	5	5	5	5	5
	J	5	5	5	0	0	0	0	5	5	5	5	5	5	5	5	5
	V	5	5	5	0	0	0	0	5	5	5	5	5	5	5	5	5
4	L	5	5	5	5	0	0	0	0	5	5	5	5	5	5	5	5
	M	5	5	5	5	5	5	5	0	0	0	0	5	5	5	5	5
	M	5	5	5	5	0	0	0	0	5	5	5	5	5	5	5	5
	J	5	5	5	5	5	5	5	0	0	0	0	5	5	5	5	5
	V	5	5	5	5	0	0	0	0	5	5	5	5	5	5	5	5
5	L	5	5	5	5	5	5	5	1	1	1	1	1	2	5	5	5
	M	5	5	5	5	5	5	5	1	1	1	1	1	2	5	5	5
	M	5	5	5	5	5	5	5	1	1	1	1	1	2	5	5	5
	J	5	5	5	5	5	5	5	1	1	1	1	1	2	5	5	5
	V	5	5	6	5	5	5	5	1	1	1	1	1	2	5	5	5

Fig. 5.6: Preferencias horarias de los monitores.

Horas posibles por monitor y preferencias diarias.

En la pestaña *"otras preferencias"*, nos reservamos la posibilidad de eliminar alguna hora de trabajo diaria por monitor ($u2$), véase Figura 5.7, en este caso no se consideró, tomando todas iguales a 1, pero se podrían considerar en otros casos dándole valor 0, no obstante, parece una opción razonable ya que eliminar horas posibles de trabajo reduciría la región factible, resultando más conveniente actuar sobre las preferencias ($P2$) como se hizo finalmente.

En la misma pestaña se considera la introducción de preferencias de días de trabajo (véase Figura 5.8), en este caso todas iguales a 1. Esta preferencia nos permitiría

		HORAS POSIBLES POR MONITOR U2															
		7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
MONITOR	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Fig. 5.7: Horas posibles diarias por monitor.

eliminar algún día de trabajo de cierto monitor introduciendo un valor más alto en el día correspondiente. La rapidez de ejecución del programa nos permitiría comprobar fácilmente como afectaría al resto de monitores y poder tomar una decisión en consecuencia.

		PREFERENCIAS DIA				
	(P1)	L	M	M	J	V
MONITOR	1	1	1	1	1	1
	2	1	1	1	1	1
	3	1	1	1	1	1
	4	1	1	1	1	1
	5	1	1	1	1	1
	6	1	1	1	1	1
	7	1	1	1	1	1
	8	1	1	1	1	1
	9	1	1	1	1	1
	10	1	1	1	1	1
	...	...	...	...	...	...

Fig. 5.8: Preferencias de días de trabajo por monitor.

Criterios de resolución.

Finalmente en la pestaña *"Criterio"*, se introducirán los parámetros que nos permitirán resolver el problema de 12 formas diferentes, según los criterios de resolución explicados en la sección 5.2 (véase Figura 5.9).

RESOLUCION	1	**1**-1ºAct **2**- 1º Sala **3**-Junto
TIPO HORARIO	1	**1** - Normal **3** - Vacaciones
CRITERIO	1	**1** - (1º ACT) **2** - (1º Preferencia horaria)

Fig. 5.9: Criterios de resolución.

5.5. Comprobación de datos.

Antes de ejecutar el programa es necesario comprobar ciertos aspectos, importantes para obtener un resultado lo más satisfactorio posible:

En primer lugar comprobar que coinciden la oferta y la demanda de horas, para ello en la hoja de entrada de datos se crearon unas casillas de comprobación para que al introducir los datos no exista la posibilidad de cometer un error, como intentar cubrir un horario con un número de monitores insuficientes.

Por otro lado será interesante comparar el número de monitores demandados por hora con las preferencias introducidas por los monitores. Si existiese un gran desequilibrio muchos monitores no verían cumplidas sus preferencias, pero en ningún caso afectaría al conjunto de soluciones factibles.

Para ello se optó por realizar unos diagramas de barras, donde visualmente podemos comprobar de forma muy rápida la distribución de la demanda y de las preferencias pedidas por los monitores. En la Figura 5.10 se muestra la oferta y demanda de horas diarias y en la Figura 5.11 la comparación de ambas, donde se puede observar que está bastante equilibrada.

Dichas gráficas fueron generadas en *R* y conectadas a la hoja *Excel* mediante la *librería "xlsx"*, lo que nos permite obtener automáticamente las gráficas con cada cambio en los datos de entrada.

Nótese que si existe una demanda superior a la oferta el programa asignará un monitor fuera de su rango de horas de preferencia, y por el contrario si existe mucha oferta cuando se demandan pocos monitores, no todos podrán ver cumplidas sus preferencias.

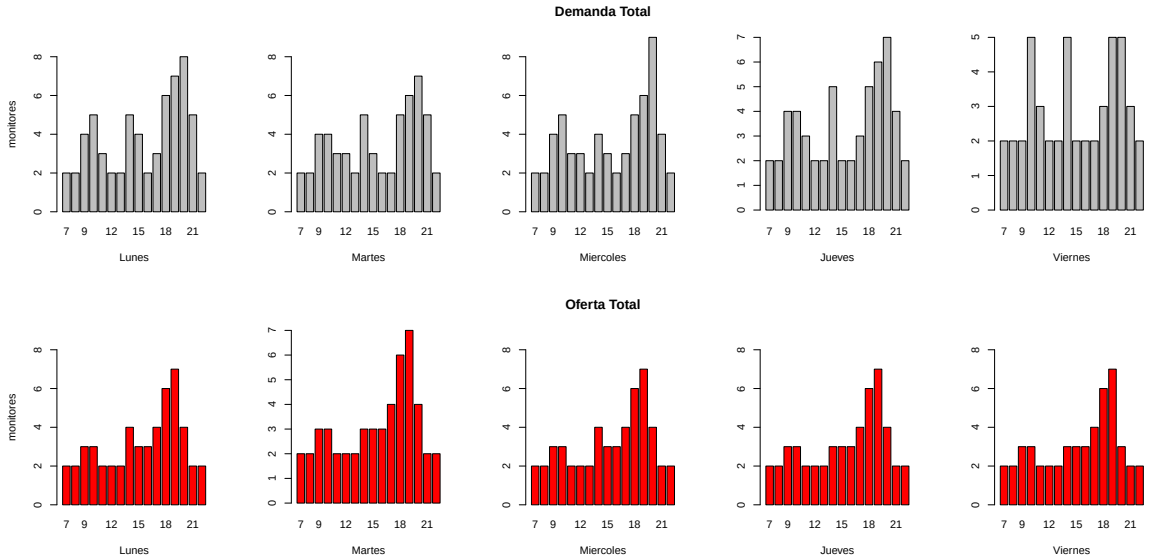

Fig. 5.10: Estudio oferta y demanda.

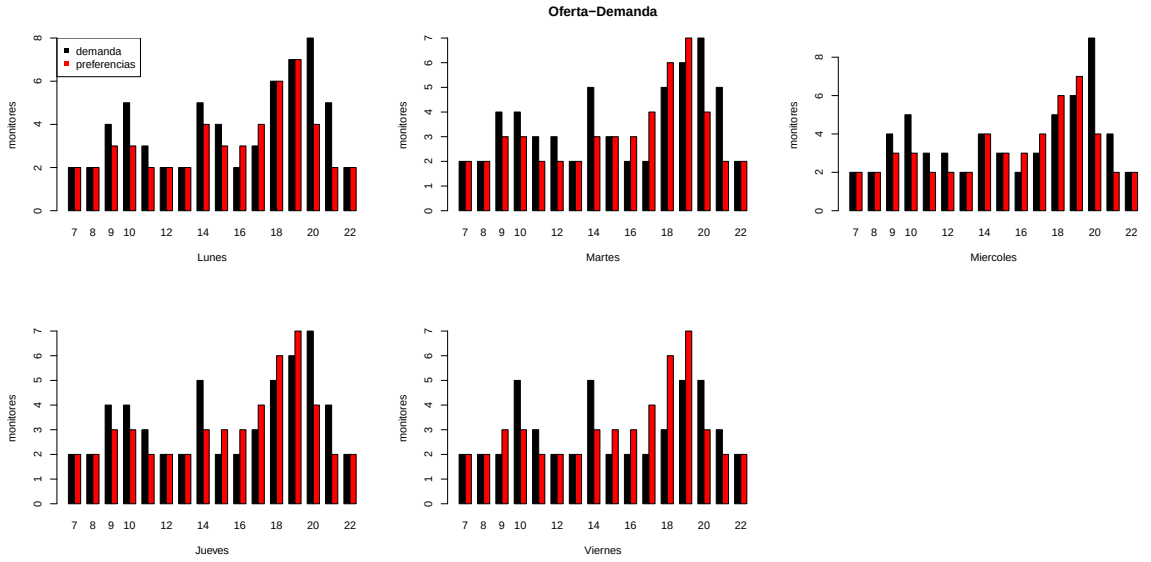

Fig. 5.11: Comparación entre oferta y demanda.

En las Figura 5.12 se muestra la oferta y la demanda, y en la Figura 5.13 su comparación, pero solo considerando las horas de sala, esta comprobación tiene cierto interés cuando se ejecuta el programa con $Res = 1$, es decir asignando primero las horas de sala.

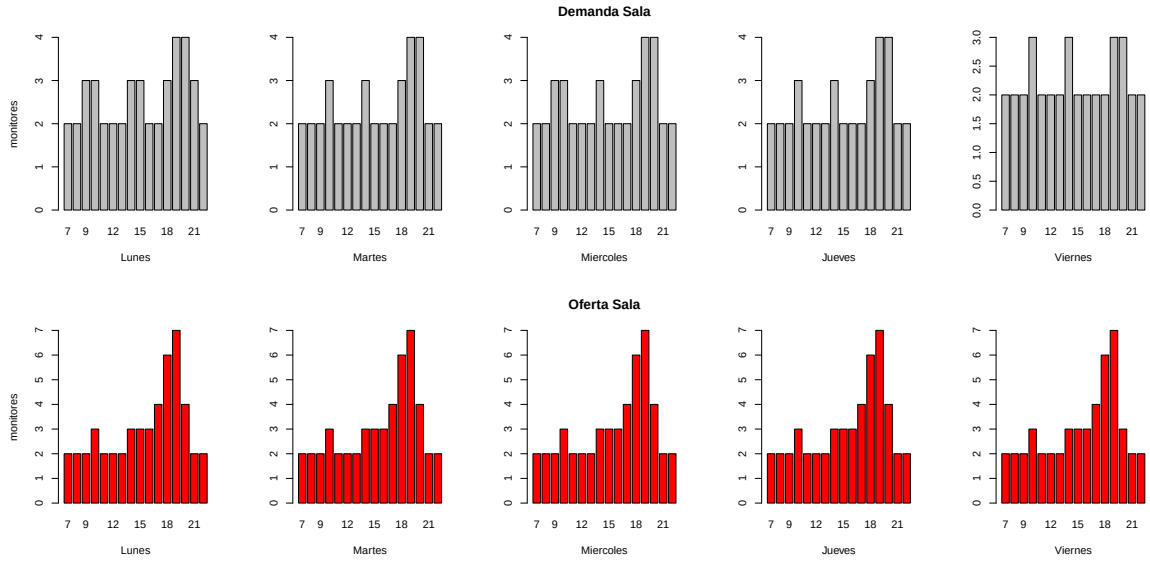

Fig. 5.12: Estudio de oferta y demanda de las horas de sala.

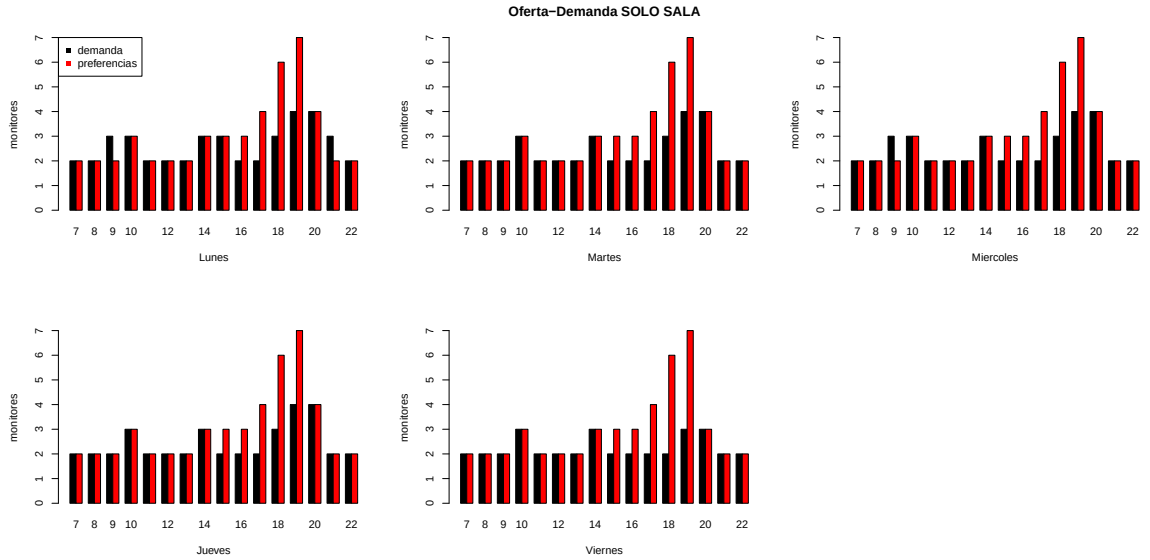

Fig. 5.13: Comparación de oferta y demanda de las horas de sala.

Estas gráficas fueron de gran utilidad en la reunión concertada con los monitores de la instalación, donde se les mostró el funcionamiento del programa, ya que rápidamente se podía ver qué horas eran más demandadas por los monitores y si coincidían con las exigencias de la instalación, lo que permitió obtener una solución más satisfactoria para todos ellos con sólo cambiar las preferencias de alguno de los monitores.

5.6. Programación.

Para la implantación, se empleó el programa *R-project* con la ayuda de las librerías "lpSolveAPI'" y "xlsx". Para su desarrollo se realizó una programación modular, creando varias funciones que agrupan las distintas tareas a desarrollar.

Debido a los tres tipos de resolución considerados, el modo de ejecución del programa debió adaptarse a cada uno de ellos. En el Anexo 1 se muestra el programa máster con la programación modular empleada.

A continuación se enumeran algunas de las funciones más importantes empleadas en el desarrollo del programa:

cargadatos(fichero,d); esta función a la que se le pasa el nombre del fichero con los datos de entrada y el número de días considerado d, carga todos los datos y parámetros necesarios para la resolución del problema.

gestiondatosP1(datos); función diseñada para el paso 1, que introduciendo todos los datos del problema prepara los vectores que posteriormente serán utilizados por el *Solver*.

gestiondatosP2(datos); función diseñada para el paso 2, al igual que *gestiondatosP1* prepara los vectores que posteriormente serán utilizados por el *Solver*, pero eliminando previamente las horas asignadas en el paso 1.

gestiondatosP3(datos); función diseñada exclusivamente para el tipo de resolución $Res = 3$, al igual que las anteriores prepara los vectores que posteriormente serán utilizados por el *Solver*, pero realizando la asignación en un solo paso.

pfcm(u,l,e,c,o,t); función que resuelve el PFCM, a la cual se le introducen los vectores creados con la correspondiente función *gestiondatos*. Interiormente emplea diferentes funciones del paquete *lpSolveAPI*, devolviendo la solución en forma vectorial.

prepsalida(pfcm,m,d,h,a); función que transforma la salida vectorial obtenida con *pfcm* en distintas formas matriciales.

salidaT(solucionF1,solucionF2,solucionF3,Res); función que imprime la salida en una hoja *Excel*, partiendo de las soluciones matriciales proporcionadas por *prepsalida*.

5.7. Velocidad computacional.

Una de las principales preocupaciones cuando nos planteamos resolver este problema con un número tan grande de variables era comprobar si el tiempo de ejecución sería muy elevado, lo que restaría manejabilidad al programa. Para ello se realizaron diversas pruebas de velocidad de computación con redes de flujo sencillas pero con un número de variables similar, obteniendo tiempos de computación sorprendentemente bajos en comparación a lo que se podría imaginar a priori.

El tiempo de ejecución del programa actualmente es de $90''$ en los casos $Res = 1$ y 2, en los cuales se resuelven un mínimo de 4 veces el PFCM (el caso más lento), lo que nos sigue sorprendiendo, no olvidemos que estamos manejando 33.120 variables y 73.568 restricciones.

Este tiempo de resolución relativamente bajo nos permite hacer cambios en los valores de entrada y comprobar rápidamente como afectan a la solución obtenida, permitiéndonos obtener distintas soluciones con diferentes criterios y quedarnos con la más adecuada en cada momento.

Capítulo 6

Resultados.

En este capítulo se muestran los distintos formatos de salida diseñados en el programa, así como los resultados obtenidos para dos de las combinaciones posibles de criterios de resolución.

6.1. Formatos de salida.

Como formato de salida se optó nuevamente por una hoja *Excel*, *"salidaT.xlsx"*, que se obtiene directamente desde *R* con la ayuda de la librería *xlsx*. En ella se crearon las siguientes pestañas que permiten representar la solución en distintos formatos adaptados a las exigencias de la instalación:

- *"horas día"* ; muestra las horas de trabajo diario que le corresponden a cada monitor, véase Figura 6.1.
- *"dist horas"*; muestra la distribución de horas a lo largo de la jornada laboral para cada monitor, véase Figura 6.2.
- *'dist monitores"*; muestra la distribución horaria y de actividades que deberá impartir cada monitor diariamente, en la Figura 6.3 se muestra el horario del lunes obtenido para los llamados *monitores A*. Esta salida se creó especialmente para controlar que monitores deberán estar en la instalación, a cada hora del día.
- *"dist monitores2"*; muestra igualmente la distribución horaria y de actividades que deberá impartir cada monitor diariamente pero configurada de distinta forma, pensada para que sea entregada a cada monitor (véase Figura 6.4).

MONITOR	L	M	M	J	V	SUMA
1	6	7	6	6	5	30
2	6	6	6	6	6	30
3	5	5	5	4	4	23
4	4	5	4	5	4	22
5	5	5	6	4	2	22
6	6	4	5	3	5	23
7	5	4	5	5	3	22
8	6	6	6	6	6	30
9	4	4	4	4	3	19
10	4	5	5	5	4	23
11	0	0	1	0	0	1
12	1	1	1	0	0	3
13	2	0	1	1	1	5
14	1	0	0	0	1	2
15	0	1	0	1	0	2
16	1	0	0	0	0	1
17	1	0	1	1	1	4
18	0	0	1	1	1	3
19	1	2	0	0	0	3
20	2	0	0	2	0	4
21	0	0	0	0	0	0
22	1	0	0	0	0	1
23	1	2	2	1	1	7

Fig. 6.1: Horas de trabajo diario por monitor.

			DISTRIBUCION HORARIA														
			7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
MONITOR	1	L	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0
		M	1	1	1	1	0	0	0	0	0	0	0	1	1	1	0	0
		M	1	1	1	1	1	0	0	0	0	0	0	1	0	0	0	0
		J	1	1	1	1	0	0	0	0	0	0	0	1	0	1	0	0
		V	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0
	2	L	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0
		M	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0
		M	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0
		J	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0
		V	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0	0
	3	L	0	0	0	1	1	1	1	0	0	0	0	0	0	1	0	0
		M	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0
		M	0	0	0	1	1	1	1	0	0	0	0	0	0	1	0	0
		J	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
		V	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	4	L	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
		M	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0	0
		M	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
		J	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0	0
		V	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

Fig. 6.2: Distribución de horas diarias.

		MONITOR									
		1	2	3	4	5	6	7	8	9	10
LUNES	7	SALA	SALA								
	8	SALA	SALA								
	9	SALA	SALA				SALA				
	10	CYCLING	SALA	SALA			SALA				
	11			SALA	SALA						
	12			SALA	SALA						
	13			SALA	SALA						
	14				SALA	SALA		SALA			
	15					SALA		SALA	SALA		
	16					SALA			SALA		
	17						SALA		SALA		
	18	CYCLING	AQUA G.			SALA	MET		SALA		SALA
	19	SALA	ZUMBA				CYCLING	SALA	SALA	SALA	
	20			RUNING			SALA	CYCLING	SALA	SALA	SALA
	21					SALA		SALA		SALA	MET
	22									SALA	SALA

Fig. 6.3: Distribución horaria de actividades.

	MONITOR 1					MONITOR 2					MONITOR 3				
	L	M	M	J	V	L	M	M	J	V	L	M	M	J	V
7	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA					
8	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA					
9	SALA	CYCLING	SALA	CYCLING	SALA	SALA	AEROSTYLE	SALA	SALA	SALA					
10	CYCLING	SALA	AEROSTEP	MET	CYCLING	SALA	FITNNES	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA
11			MET								SALA	SALA	SALA	SALA	SALA
12											SALA	SALA	SALA	SALA	SALA
13											SALA	SALA	SALA	SALA	SALA
14												RUNING			
15															
16															
17															
18	CYCLING	SALA	FITNNES	SALA	SALA	AQUA G.	SALA	AQUA G.	AQUA G.	AEROSTEP					
19	SALA	CYCLING				ZUMBA	MET	CYCLING	SALA	MET					
20		SALA		SALA							RUNING		RUNING		
21															
22															

Fig. 6.4: Distribución horaria de actividades.

6.2. Análisis de la solución.

Debido al gran número de resoluciones posibles en esta sección sólo nos centraremos en dos de ellas, las que según la dirección, parecen más interesantes para este problema en concreto.

Soluciones obtenidas para la resolución 1, tipo 1 y criterio 1.

En primer lugar se analizan las soluciones obtenidas para $Res = 1$,$T = 1$ y $Cri = 1$, criterios elegidos por la dirección que mejor se adaptan a las condiciones requeridas por la instalación.

La Figura 6.4 mostrada en la sección anterior, reflejaba los resultados obtenidos con dichos parámetros, donde podemos comprobar la similitud con las preferencias horarias introducidas ($P2$) por estos monitores (véase 5.6), además de impartir las

actividades elegidas como primera opción. Estos resultados nos dejan totalmente satisfechos ya que la mayoría de los monitores ven cumplidas sus prioridades.

Cabe destacar de los resultados obtenidos, por ejemplo, el horario de los monitores "4" y "8", véase Figura 6.5, en la que se consiguen intercalar sus horas de sala en días consecutivos, según sus preferencias. Dicha solución sería muy complicada de obtener manualmente ya que dichos monitores tienen una carga horaria diferente y deseaban trabajar en días alternos de mañana y tarde.

Como vimos en la Sección 4.4 al ejecutar el programa por pasos no nos aseguramos encontrar el óptimo, en este caso se obtuvo un valor en el objetivo de 857, mientras que si lo ejecutamos en un único paso ($Res = 3$) obtenemos 845. Efectivamente la ejecución en dos pasos no asegura el óptimo pero si un valor muy cercano.

	MONITOR 4					MONITOR 8				
	L	M	M	J	V	L	M	M	J	V
7										
8										
9							SALA		SALA	
10							SALA		SALA	
11	SALA		SALA		SALA		SALA		SALA	
12	SALA		SALA		SALA		SALA		SALA	
13	SALA		SALA		SALA		SALA		SALA	
14	SALA	SALA	SALA	SALA	SALA		SALA		SALA	
15		SALA		SALA		SALA		SALA		SALA
16		SALA		SALA		SALA		SALA		SALA
17		SALA		SALA		SALA		SALA		SALA
18						SALA		SALA		SALA
19						SALA		SALA		SALA
20		SWIMMING		SWIMMING		SALA		SALA		SALA
21										
22										

Fig. 6.5: Distribución horaria de los monitores 4 y 8.

Soluciones obtenidas para la resolución 1, tipo 3 y criterio 1.

Ya que uno de los objetivos del programa era la confección de horarios en la etapa vacacional de los monitores, nos parece interesante mostrar las salidas obtenidas cuando se simula el horario, por ejemplo, con la falta de dos de los *monitores A*.

Antes de ejecutar el programa es necesario asegurarse de que los monitores que van a sustituir a los de vacaciones puedan cubrir las horas impartidas por éstos, para variar lo menos posible las horas asignadas al resto de monitores, por ejemplo si los monitores de sustitución son del tipo B se les debe permitir impartir horas de sala.

En este caso se simuló que los monitores que toman vacaciones son el “4” y el “8” (*monitores A*), y los que los tendrían que sustituir son los *monitores B* “12” y “13”, permitiendo a estos últimos que impartan horas de sala.

En la Figura 6.6 se puede observar la distribución de las horas asignadas a los mismos monitores que se mostraron en la figura 6.4, donde se comprueba que continúan impartiendo sus actividades en la franja horaria preferida que es lo que realmente se buscaba; no provocar un trastorno al resto de monitores de la instalación cuando alguno de ellos tomara vacaciones.

	MONITOR 1					MONITOR 2					MONITOR 3				
	L	M	M	J	V	L	M	M	J	V	L	M	M	J	V
7	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA					
8	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA	SALA					
9	SALA	CYCLING	SALA	CYCLING	SALA	SALA	AEROSTYLE	SALA	SALA	SALA					
10	CYCLING	FITNNES	SALA	MET	CYCLING	SALA	SALA	AEROSTEP	SALA	SALA	SALA	SALA	SALA	SALA	SALA
11	SALA				SALA			MET			SALA	SALA	SALA	SALA	SALA
12											SALA	SALA	SALA	SALA	SALA
13											SALA	SALA	SALA	SALA	SALA
14												RUNING			
15															
16															
17															
18	AQUA G.	AQUA G.	AQUA G.	SALA		MET	SALA	FITNNES	SALA	AEROSTEP					
19		SALA	SALA		SALA	ZUMBA	MET		CYCLING	CYCLING					
20			AQUA A.								RUNING		RUNING		
21															
22															

Fig. 6.6: Distribución horaria y de actividades en periodo vacacional.

6.3. Validación de los resultados.

Las soluciones mostradas en la sección anterior son las producidas automáticamente por el programa, pero éstas podrían ser modificadas fácilmente por el usuario si lo creyese conveniente, o si algunos monitores deseasen intercambiarse alguna hora. Para ello se destina la pestaña *“dist monitores”* en la que se pueden realizar los cambios pertinentes, que con la ayuda de otro pequeño programa creado en *R*, el cual lee dicha hoja y la transforma en la salida definitiva con el formato de la pestaña *“dist monitores2”*.

Otra opción que nos permite hacer es cambiar la hora de inicio de una actividad, siempre que lo permita el horario del monitor.

Pensando en la posibilidad de introducir las horas de inicio de actividades cada media hora, habría que intentar que dichas actividades sean asignadas a monitores tipo B, manipulando de forma adecuada las preferencias de estos para que le sean asignadas prioritariamente.

Capítulo 7

Limitaciones y mejoras.

La limitación más importante en el momento actual de desarrollo del programa es que las unidades de flujo que salen de la red tienen que tener el mismo valor, es decir, no se pueden demandar actividades que duren una hora y otras que duren dos. Para controlar que estas actividades de dos horas fueran impartidas por el mismo monitor habría que incluir restricciones a mayores en el problema, que de momento no está resuelto debido al tamaño del problema considerado. Este es el mismo tipo de problema que nos encontramos cuando existe la restricción de rigidez impuesta por las horas de sala y actividades.

Para resolver este problema de rigidez se nos ocurrió recientemente considerar un diseño del grafo alternativo en el que se duplica el grafo para sala y actividades como se puede ver en la Figura 7.1. Este diseño nos permite controlar exactamente el número de horas de sala y actividades que impartirá cada monitor, pero es necesario incluir un conjunto de restricciones que aseguren que un monitor no imparte a la vez una hora de sala y actividad. Estas restricciones serían del tipo $f_{adh} + f_{sdh} = 1$, es decir que el día d a la hora h solo imparta una de ellas. El implementar esta forma de resolución no está resuelta por ahora ya que debemos asegurarnos que no se pierde la propiedad de unimodularidad al introducir estas restricciones externas al problema de flujo, trabajo que nos proponemos realizar en el futuro.

De lo aprendido a lo largo del desarrollo del trabajo y del conocimiento adquirido del funcionamiento de este tipo de instalaciones, otra de las mejoras que se podrían introducir al programa es la adaptación a horarios en donde se consideren medias

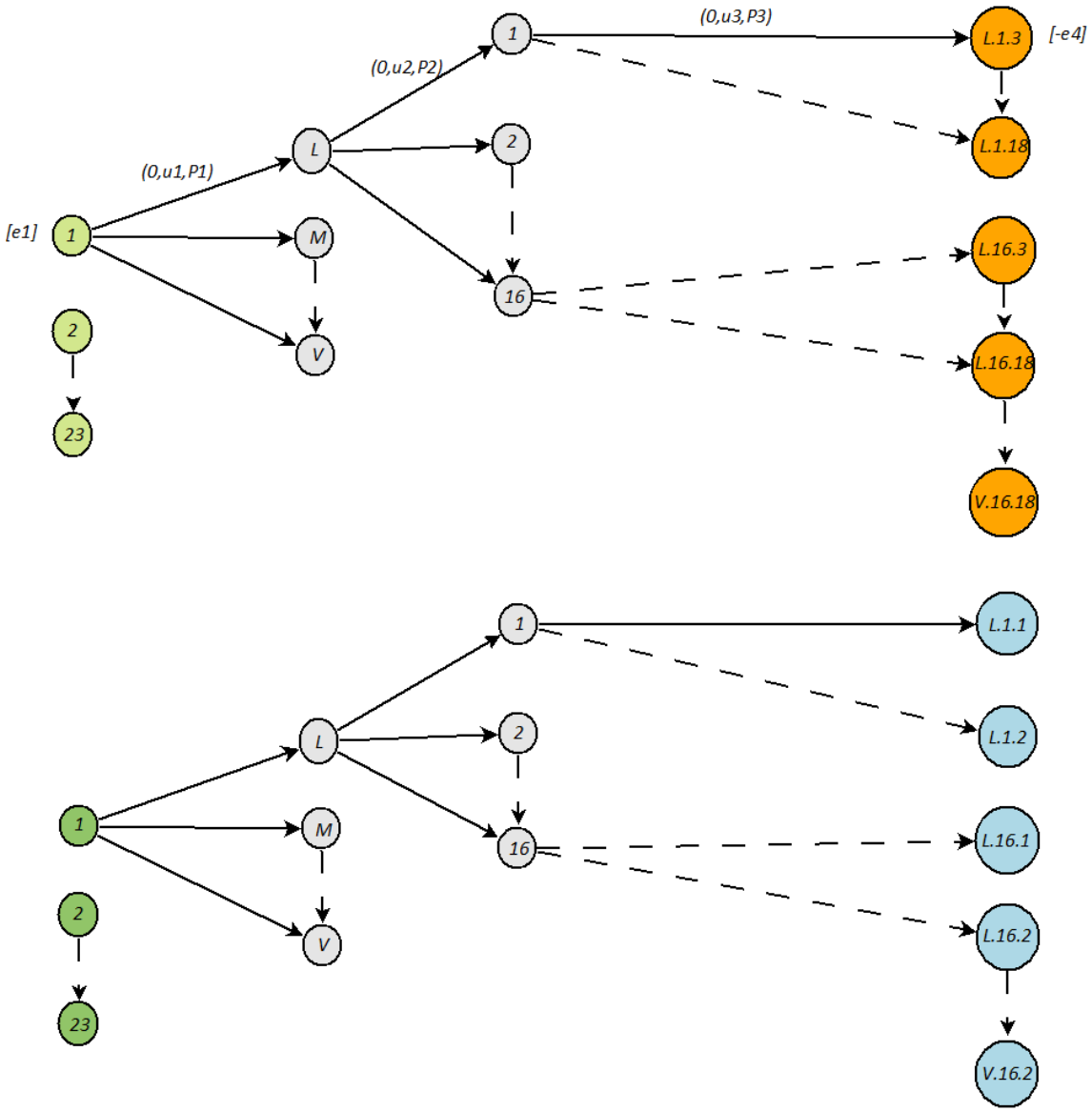

Fig. 7.1: Grafo alternativo.

horas, es decir, que las actividades puedan empezar cada media hora y no sólo a las horas en punto. Esta mejora conllevaría duplicar el número de variables y por tanto de todo el problema, con lo que se superaría la capacidad de memoria del ordenador empleado. Para resolverlo se contemplan dos posibilidades:

- Emplear una programación alternativa con la inclusión de estructuras *"sparse"*, que consisten en almacenar únicamente las entradas de las matrices que son distintas de cero. Como en todo PFCM cada columna de la matriz de restricciones tiene sólo dos entradas distintas de cero, permitiría reducir dramáticamente el espacio requerido para almacenar el problema.

- Emplear el paquete de *R "ffbase"* diseñado para trabajar con gran cantidad de datos. Cuestiones que nos proponemos probar próximamente.

Como en el caso del problema de rigidez, sería necesario incluir un conjunto de restricciones externas al problema de flujo que asegurasen que los monitores impartiesen dos medias horas consecutivas. Siendo imprescindible el uso de un programa específico de programación entera, como por ejemplo *"Gurobi"*, para resolver el problema con esta cantidad de variables.

Otra de las inquietudes que quedan por resolver es aplicar el programa a otro tipo de instalaciones como podría ser el horario de clases de un colegio o a los horarios del máster, adecuándolos según las preferencias horarias de los profesores.

Finalmente y debido a los buenos resultados obtenidos con el programa, nos gustaría incluir una interface gráfica que unificase la entrada de datos, la resolución y la salida, facilitando su uso a los posibles usuarios y así poder implantarlo definitivamente en este tipo de instalaciones.

Capítulo 8

Conclusiones.

Este trabajo ha permitido comprobar que un problema de automatización de horarios se puede plantear como un problema de flujo en redes, obteniendo un conjunto de soluciones factibles que además se pueden optimizar en función de las preferencias horarias y de actividades de los empleados de la empresa.

Una virtud importante del programa es la rapidez en la obtención de las soluciones, lo que permite comprobar como varía la solución si introducimos un cambio en cualquiera de los parámetros o preferencias considerados, permitiendo por ejemplo, que uno de los monitores pidiese tener un día libre a la semana o unas horas diarias por motivos personales.

Por otro lado, cabe destacar el buen comportamiento del programa cuando este se ejecuta en forma vacacional, que era uno de los objetivos iniciales considerados, ya que las soluciones obtenidas siguen manteniendo prácticamente invariable los horarios de los monitores en este periodo, algo reiteradamente demandado por todos ellos.

El programa además podría adaptarse fácilmente a otra instalación de características similares aunque varíen el número de monitores, horas, días y actividades y adecuarse a diferentes criterios según las características particulares de cada instalación, gracias a los distintos parámetros introducidos en el programa.

Con el mismo desarrollo conceptual el programa podría adaptarse a otro tipo de instalaciones como colegios o universidades.

Bibliografía.

Gotlieb, C. C. (1963). *"The Construction of Class-Teacher Timetable"*. Proc. IFIP Congress 62, 73-77, North Holland.

A. Schaerf (1999).*"A Survey of Automated Timetabling "*. Artificial Intelligence 13, 87-127.

E.K. Burke (2008).*A survey of search methodologies and automated system development for examination timetabling.*J Sched (2009) 12, 55?89.

Ostermann, R., and de Werra, D. (1983). *Some experiments with a timetabling system.* OR Spektrum 3: 199-204.

De Werra, D. (1985). *An introduction to timetabling.* European Journal of Operational Research 19: 151- 162.

Hitchcock, F. L. (1941). *The Distribution of a Product from Several Sources to Numerous Localities.* Journal of Mathematical Physics. American Institute of Physics (AIP) 20, pp.224-230.

Dantzig, G. B. (1951). *Aplication of the Simplex Method to a Transportation Problem.* Monografía n.º 13 de la Cowles Commision. John Wiley, New York.

Ford, L. R.; Fulkerson, D. R. (1962). *Flows in Network.* Princeton University Press, N. J.

Julio González Díaz (2012-2013). Apuntes de la asignatura *"Programación lineal y entera"*. Máster interuniversitario en técnicas estadísticas. USC, UDC y UVigo.

Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, New Jersey: PrenticeHall.

Anexo.
Programa máster.

```
#########################################################################
########################    HORARIOS    ################################
#########################################################################

library(lpSolveAPI)
library(xlsx)

# Cargar fichero con todas las funciones del programa.

source("funcioneshorarios2.R")

############################  Datos  ###################################

fichero="HORARIOS.xlsx"
d=5  # nº de días

############################ CARGAR DATOS  ############################

datos<-cargadatos(fichero,d)

Cri<-datos$C
Tipo<-datos$T
Res<-datos$R
m<-datos$m
h<-datos$h
a<-datos$a

# SI RES=1 primero distribuimos ACT y después SALA
# SI RES=2 primero distribuimos SALA y después ACTIVIDADES
# SI RES=3 distribuimos SALA y ACTIVIDADES a la vez

# SI CRI=1 PREFERENCIAS ACTIVIDADES
# SI CRI=2 PREFERENCIAS HORAS

#########################################################################
######################  RESOLUCIÓN 1 Y 2 ################################
#########################################################################

if(Res==1|Res==2){

############################  PASO 1 ###################################

vectordatos<-gestiondatosP1(datos)
```

```
u<-vectordatos$u
l<-vectordatos$l
e<-vectordatos$e
c<-vectordatos$c
o<-vectordatos$o
t<-vectordatos$t
P3<-vectordatos$P3
c1<-vectordatos$c1
c2<-vectordatos$c2
c3<-vectordatos$c3
################################## FUNCIÓN pfcm ########################

pfcm1<-pfcm(u,l,e,c,o,t)

################################  SALIDA  ############################

salida1<-prepsalida(pfcm1,m,d,h,a)

solucionS1<-salida1$solucionS1
solucionS2<-salida1$solucionS2
solucionS3<-salida1$solucionS3

########################## POSTPROCESADO 1 ##########################

# SI RES =1 (1° ASIGNAMOS ACTIVIDADES) REVISAMOS NO REPETIR 3 CYCLING

if (Res==1) {

P3aux<-P3
ii=dim(solucionS3)[1]   # n° de filas

j=3   # n° de actividad (cycling)

for (i in 3:ii){
if (solucionS3[i-2,j]==1 & solucionS3[i-1,j]==1) {P3aux[i,j]<-20}
}
ii=dim(P3aux)[1]
jj=dim(P3aux)[2]
icont=1
for (i in 1:ii){
for (j in 1:jj){
c3[icont]=P3aux[i,j]
icont=icont+1
}}
```

```
c<-c(c1,c2,c3)

################################## FUNCIÓN pfcm ######################

pfcm2<-pfcm(u,l,e,c,o,t)

################################  SALIDA  ############################

pfcm2$objetivo

salida1<-prepsalida(pfcm2,m,d,h,a)

solucionS1<-salida1$solucionS1
solucionS2<-salida1$solucionS2
solucionS3<-salida1$solucionS3
}

# SI RES NO ES IGUAL A 1 SEGUIMOS

############################  PASO 2 ################################

vectordatos2<-gestiondatosP2(datos,solucionS2)

u<-vectordatos2$u
l<-vectordatos2$l
e<-vectordatos2$e
c<-vectordatos2$c
o<-vectordatos2$o
t<-vectordatos2$t

################################## FUNCIÓN pfcm ######################

pfcm3<-pfcm(u,l,e,c,o,t)

################################  SALIDA  ############################

salida<-prepsalida(pfcm3,m,d,h,a)

solucionS12<-salida$solucionS1
solucionS22<-salida$solucionS2
solucionS32<-salida$solucionS3

########################## POSTPROCESADO 2 ###########################
```

```
# SI RES =2 (2° ASIGNAMOS ACTIVIDADES) REVISAMOS NO REPETIR 3 CYCLING

if (Res==2) {

P3aux<-P3

ii=dim(solucionS3)[1]   # n° de filas

j=3   # n° de actividad (cycling)

for (i in 3:ii){
if (solucionS3[i-2,j]==1 & solucionS3[i-1,j]==1) {P3aux[i,j]<-20}
}

ii=dim(P3aux)[1]
jj=dim(P3aux)[2]

icont=1
for (i in 1:ii){
for (j in 1:jj){
c3[icont]=P3aux[i,j]
icont=icont+1
}}

c<-c(c1,c2,c3)

################################## FUNCIÓN pfcm ######################

pfcm4<-pfcm(u,l,e,c,o,t)

################################  SALIDA  ############################

salida<-prepsalida(pfcm4,m,d,h,a)

solucionS12<-salida$solucionS1
solucionS22<-salida$solucionS2
solucionS32<-salida$solucionS3
}

## SINO SEGUIMOS

solucionF1<-solucionS12+solucionS1
solucionF2<-solucionS22+solucionS2
solucionF3<-solucionS32+solucionS3
```

```
salidaT(solucionF1,solucionF2,solucionF3,Res)

pfcm1$objetivo
pfcm2$objetivo
pfcm3$objetivo
pfcm4$objetivo
}

#####################################################################
######################   RESOLUCIÓN 3    ############################
#####################################################################

if(Res==3){

vectordatos<-gestiondatosP3(datos)

u<-vectordatos$u
l<-vectordatos$l
e<-vectordatos$e
c<-vectordatos$c
o<-vectordatos$o
t<-vectordatos$t
P3<-vectordatos$P3
c1<-vectordatos$c1
c2<-vectordatos$c2
c3<-vectordatos$c3

################################## FUNCIÓN pfcm ####################

pfcm1<-pfcm(u,l,e,c,o,t)

################################  SALIDA  ##########################

salida1<-prepsalida(pfcm1,m,d,h,a)

solucionS1<-salida1$solucionS1
solucionS2<-salida1$solucionS2
solucionS3<-salida1$solucionS3

############################ POST 1  ##############################
P3aux<-P3

ii=dim(solucionS3)[1]   # nº de filas
```

```
j=3   # nº de actividad (cycling)

for (i in 3:ii){
if (solucionS3[i-2,j]==1 & solucionS3[i-1,j]==1) {P3aux[i,j]<-20}
}

ii=dim(P3aux)[1]
jj=dim(P3aux)[2]

icont=1
for (i in 1:ii){
for (j in 1:jj){
c3[icont]=P3aux[i,j]
icont=icont+1
}}

c<-c(c1,c2,c3)

############################## FUNCIÓN pfcm ##########################

pfcm2<-pfcm(u,l,e,c,o,t)

############################  SALIDA  ##############################

salida1<-prepsalida(pfcm2,m,d,h,a)

solucionF1<-salida1$solucionS1
solucionF2<-salida1$solucionS2
solucionF3<-salida1$solucionS3

salidaT(solucionF1,solucionF2,solucionF3,Res)
}

Res;pfcm1$objetivo;pfcm2$objetivo;pfcm3$objetivo;pfcm4$objetivo
```