Haciendo horarios

Juan Félix Avila Herrera.
Escuela de Informática, Universidad Nacional.
javila@una.ac.cr

Resumen

En este trabajo se aborda el problema de la construcción de horarios mediante técnicas de Investigación de Operaciones. El modelo propuesto está basado en teoría de Flujo de Redes. Se presenta además un software desarrollado por el autor y denominado $\mathcal{H}$orario *que es capaz de generar (tanto manual como automáticamente) un horario para un centro educativo en donde varios profesores imparten varias materias. Se deben respetar las disponibilidades de tiempo de cada docente de modo que se reduzca la cantidad de lecciones que queden sin asignación.*

Introducción

El problema de confeccionar un horario respetando las disponibilidades de tiempo de los profesores es una tarea que tal vez a algunas personas les pueda parecer sencilla, pero quienes han pasado por esta experiencia, saben que proceso no es para nada trivial y que además su dificultad crece (probablemente exponencialente), conforme se incrementa la cantidad de profesores y grupos por atender. Confeccionar un horario es un proceso que puede abordarse desde el punto de vista de la asignación de recursos a necesidades. En este caso los recursos lo constituyen los docentes con sus disponibilidades de tiempo. Las necesidades, por otro lado son los grupos que desean recibir cierta cantidad de lecciones con algunos de esos profesores. El problema para instancias pequeñas se resuelve generalmente en forma mental y generalmente ni si quiera se piensa si la solución encontrada es óptima o no. La solución ingenua (que funciona para instancias pequeñas) es probar todas las posibilidades y quedarse con la mejor. Es decir, consideremos todas las posibilidades del profesor i mientras fijamos las de los demás. Si suponemos que el centro educativo cuenta con 50 profesores y que cada uno debe atender 30 lecciones, el número total de posibilidades a considerar es de

$$30^{50}$$

que en algunas calculadoras de bolsillo producirá un *overflow* o desbordamiento.

Redes de transporte

Una red de transporte es una herramienta que nos permite modelar problemas sobre flujo de materiales. Imagine, por ejemplo, un producto navegando a través de un sistema desde una fuente, en donde el material es producido, hasta un destino, en donde es consumido o almacenado. La fuente produce el material y es capaz de bombearlo a una cierta velocidad fija, de suerte que los destinatarios pueden consumirlo o almacenarlo a la misma velocidad. Intuitivamente podemos pensar que el "flujo" del material en cualquier punto del sistema es la velocidad a la cual el material se mueve. De esta forma las redes de transporte pueden ser usadas para modelar líquido fluyendo a través de tuberías, partes a través de líneas de ensamblaje, corriente a través de redes eléctricas, información a través de redes de comunicación, etc. El problema de flujo máximo en redes de transporte se refiere a la consecución de la tasa o velocidad máxima a la cual el material puede ser bombeado de la fuente al destino sin violar las capacidades de los conductos intermedios.

Definición
Una *red de transporte* (o simplemente *red*) es un digrafo $G = (V, E)$ simple (i.e. sin lazos o rizos ni lados paralelos), con las siguientes propiedades:

- Existe un solo nodo en G, que solo tiene lados de salida y no tiene lados de llegada; tal nodo se denomina la *fuente*.
- Existe un solo vértice en G, que solo tiene lados de entrada y no tiene lados de salida; tal nodo se denomina el *depósito* o *destino*.
- Para cada lado $e = (i, j)$ que une digamos, los vértices v_i y v_j se asocia un número real $C_{ij} \geq 0$, que se conoce como la *capacidad* de la arista e. Formalizando un poco más este concepto, podemos pensar que existe una función de valor real $C : V \times V \mapsto [0, +\infty[$, tal que si el par $(a, b) \notin E$, entonces $C_{a,b} = C(a, b) := 0$. Dicha función se llama la *función capacidad*
- Para cualquier vértice v en G, existirá al menos un lado de entrada o un lado de salida.

No se debe confundir los conceptos de capacidad (que acabamos de definir) y flujo (que es la próxima definición), el hecho de que una tubería tenga capacidad 50, no quiere decir que ese será el flujo que transportará.

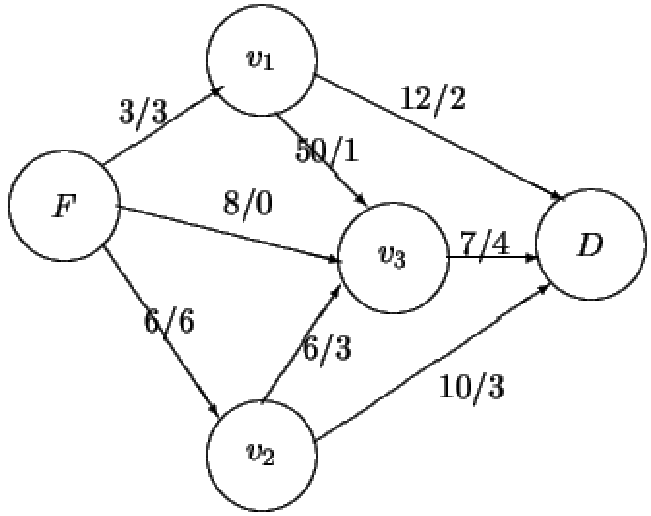
Definición
Sea G una red de transporte, y C_{ij} la capacidad de la arista (dirigida) $e = (i, j)$, definimos el *flujo* en la arista e como una cantidad real $F_{ij} \geq 0$ que satisface las siguientes características:

- (RESTRICCIÓN DE CAPACIDAD) El flujo no supera la capacidad en cada lado, i.e. $F_{ij} \leq C_{ij}$.
- (FLUJO NULO) Si entre el vértice i y el vértice j no hay arista, se define $F_{ij} = 0$.
- (CONSERVACIÓN DEL FLUJO) La cantidad de flujo que entra a un nodo (que no es ni fuente ni destino) es la misma cantidad que sale de él. Más detalladamente, para cada nodo j, que no es ni fuente ni depósito, se cumple

$$\sum_i F_{ij} = \sum_i F_{ji}.$$

Formalizando un poco más el concepto de flujo podemos pensar de éste como una función $F : V \times V \mapsto [0, +\infty[$ tal que i y j son vértices de G, entonces $F(i, j) = F_{ij}$.

Figura 1.1: F: Fuente, D: Destino.



En la Fig.1.1 utilizamos la notación c / f para indicar que la arista con capacidad c está transportando f unidades de flujo. Note que al nodo v_3 llegan 4 unidades de flujo y salen 4 unidades de flujo. Esto ilustra la propiedad de la conservación del flujo. Observe también que de la fuente salen 9 unidades de flujo y al depósito llegan 9 unidades de flujo. Esta propiedad se cumple siempre, y la registramos en el siguiente teorema. **Teorema**
Sea F un flujo sobre una red G, entonces se cumple que el flujo que sale de la fuente es igual al flujo que llega al depósito; más exactamente

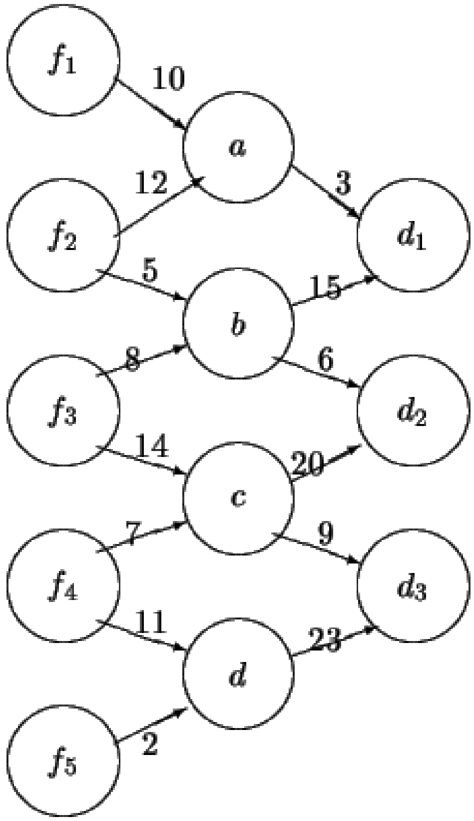
$$\sum_i F_{fi} = \sum_i F_{id},$$

en donde f es el nodo fuente y d es el nodo destino.

DEMOSTRACIÓN: El lector puede encontrar la demostración en [8].

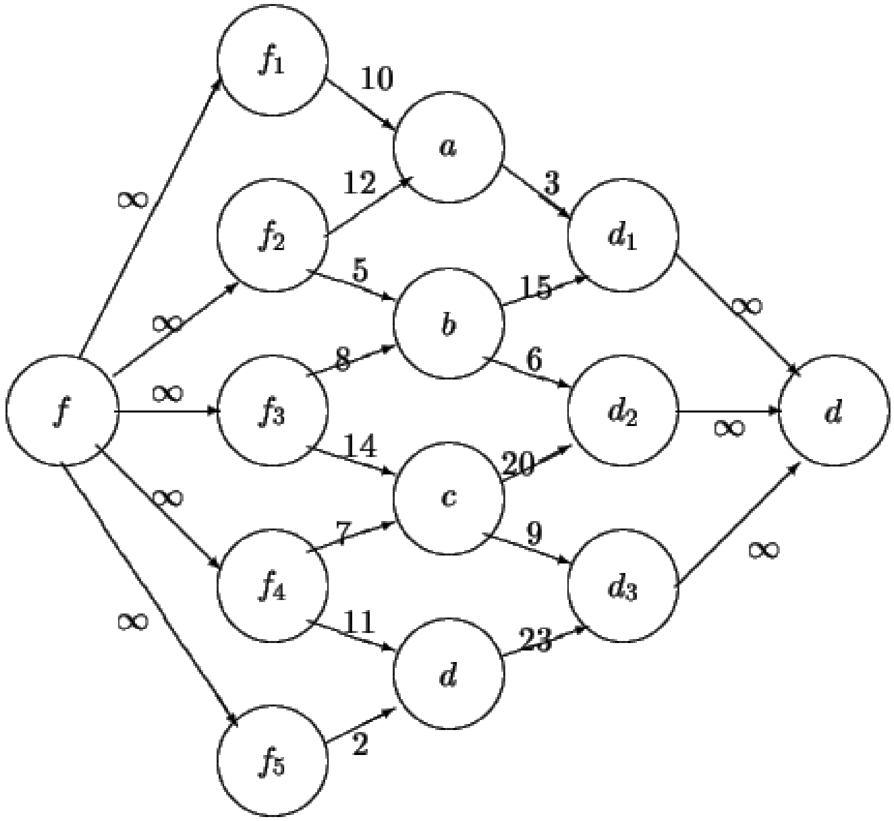
Debemos ahora retomar el problema de qué hacer en el caso de múltiples fuentes y múltiples depósitos. Considere por ejemplo el caso presentado en la Fig. 1.2, en la que se aprecian 5 fuentes y 3 depósitos.

Figura 1.2: Múltiples fuentes y múltiples depósitos



Para convertir este problema en una red de transporte (con una sola fuente y un solo depósito) podemos emplear el siguiente artificio. Agregamos dos nuevos nodos al grafo original y 8 ($8 = 5 + 3$) nuevas aristas. Con el fin de que el nuevo problema sea equivalente, definimos las capacidades de estas nuevas aristas como ∞ (por supuesto para efectos computacionales usamos un número positivo grande para emular ∞). El nodo del cual parten las nuevas aristas a los depósitos originales se llama *superdepósito*, y el otro se conoce como *superfuente*. En la Fig. 1.3 se aprecia la red resultante.

Figura 1.3: Agregamos una superfuente y un superdepósito



En virtud de esta transformación, podemos entonces llamar red inclusive a aquellas con múltiples fuentes y múltiples depósitos.

El método de Ford-Fulkerson

Para una red de transporte G, estamos interesados en obtener la forma de utilizar optimamente las "tuberías" de las que se dispone, dicho de otra forma queremos obtener la mayor cantidad de flujo que se pueda transportar por la red sin violar las capacidades predefinidas. El método de Ford-Fulkerson resuelve el problema del *flujo máximo*. Le llamamos método y no un "algoritmo" dado que permite varias implementaciones con diferentes tiempos de ejecución (para un estudio sobre tiempos de ejecución se sugiere consultar [1,4]). El método Ford-Fulkerson es iterativo, puesto que se empieza con un flujo nulo (soportado por las tuberías de la red) y luego éste se va aumentando hasta que logre el mayor flujo posible sin violar las restricciones de capacidad. Para lograr ir aumentando la cantidad de flujo transportado, en cada paso utilizaremos el concepto de *ruta aumentante* o *semivía*, la cual puede pensarse como un camino desde la fuente hasta el destino por el que se puede empujar o mandar todavía flujo. El método de Ford-Fulkerson consiste en utilizar tantas rutas aumentantes como existan. En [5], el método de Ford-Fulkerson se resume de la siguiente forma:

Método Ford-Fulkerson

1. Inicie el flujo F en 0.
2. **WHILE** exista una ruta aumentante p **DO**
3. *Aumente* el flujo F a lo largo de p
4. **RETURN** F.

Dicho en otras palabras, la idea es encontrar conductos subutilizados entre la fuente y el destino, que pueden aumentarse hasta que una restricción de capacidad detenga el aumento. En varios textos se consiguen implementaciones más o menos detalladas del método de Ford-Fulkerson, consulte la presentada en [10] (también puede consultar [5]).

El problema de las asignaciones

Podemos representar problema de las asignaciones mediante una red de transporte. Utilizamos entonces un grafo $G = (V, E)$ *bipartido*, que se define como un grafo en donde el conjunto de vértices V se puede dividir en dos subconjuntos disjuntos V_1 y V_2, de manera que cada lado $e \in E$, es incidente en un vértice de V_1 y en otro de V_2. Para convertir un grafo bipartido en una red de transporte es necesario agregar una superfuente s y un superdepósito t. En la Fig.1.4 el grafo bipartido original incluía solo los nodos p_i y t_i. El peso asignado a los nuevos lados que salen de s a los p_i es de 1, y de igual forma el peso que llega de los t_i a t es también de 1.

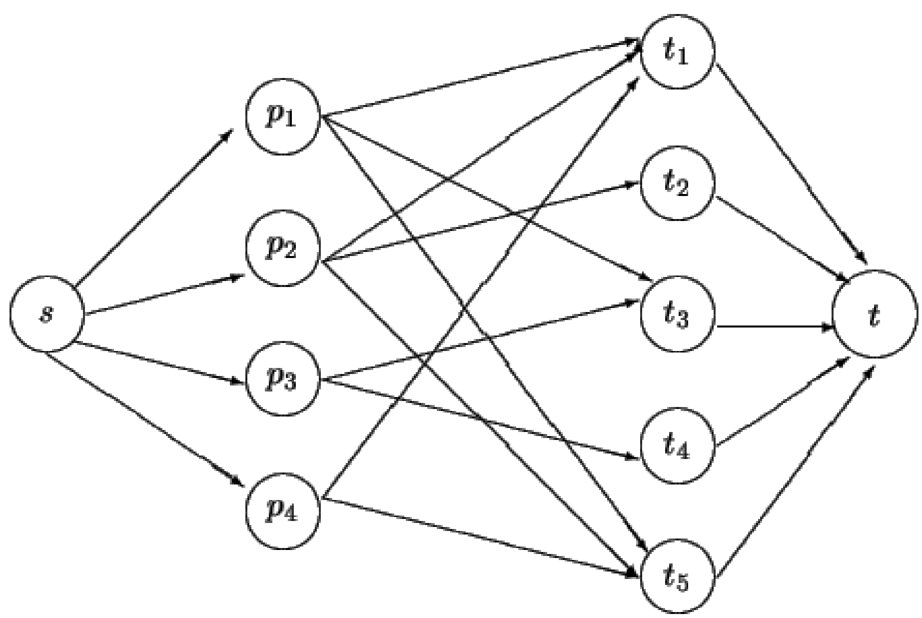


Figura 1.4: Red de transporte asignada al problema de las asignaciones.

Se debe observar que el peso asignado a cada lado en la red de la Fig.1.4, es de 1. Estamos suponiendo aquí que cada persona p_i puede atender un único trabajo t_j, de los que manifiesta en la Fig.1.4, sin embargo puede darse el caso en que una persona, por ejemplo, pueda atender dos trabajos de los tres que apetece.Para representar esta situación es suficiente asignar un peso de dos a la arista que sale de la superfuente s a la persona con esta característica. Las siguientes definiciones nos permiten formalizar aún más los conceptos que hemos venido estudiando:

Definición
Supóngase que $R = \{p_1, p_2, \cdots, p_m\}$ es un conjunto de m recursos y que

$N = \{ t_1, t_2, \cdots, t_n \}$ es un conjunto de n necesidades, entonces:

1. Una *manifestación preferencial* de a N es un conjunto $M \subseteq R \times N$.
2. La matriz A que satisface $A[i,j] = 1$ si $(i,j) \in M$ y $A[i,j] = 0$ si no, se llama *matriz de preferencias* de la manifestación preferencial M.
3. La red de transporte $G = (V, E)$, dada por:

   a. $V = R \cup N \cup \{ s \} \cup \{ t \}$, en donde s y t son respectivamente la superfuente y el superdepósito,

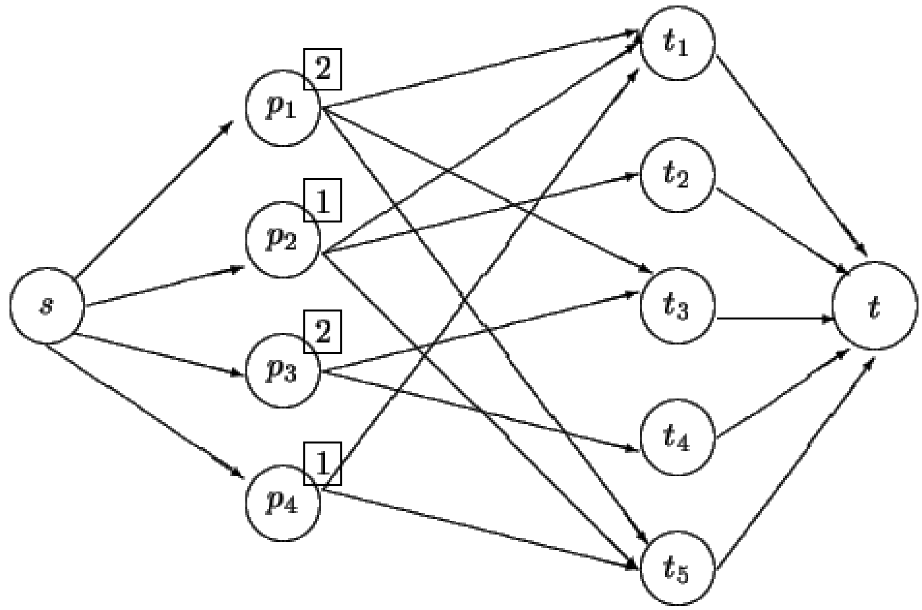
   b. $E = M \cup \{ (s, p_i) : 1 \leq i \leq m \} \cup \{ (t_i, t) : 1 \leq i \leq n \}$,

   c. y con un peso de 1 asociado a cada arista existente,se llama *red de preferencias* asociada a la manifestación preferencial M.

4. La matriz C de tamaño $(m+n+2) \times (m+n+2)$ que satisface $C[u,v] = 1$ si $(u,v) \in E$ y $C[u,v] = 0$ si no, se llama la *matriz de capacidades* de la red de preferencias.

Una vez que se ha establecido la red de preferencias para un problema, el método de Ford-Fulkerson se encargará de hacer la mayor cantidad de asignaciones posibles. En este contexto debemos observar que si el flujo entre un recurso p_i y una necesidad t_j es de 1, se concluye que ha producido una asignación entre ambas. Veamos ahora una pequeña variante del problema de las asignaciones explicado anteriormente. Supongamos que en una manifestación preferencial (como la dada en la Fig.1.4), las personas p_1 y p_3 pueden cada uno hacerse cargo de dos de sus preferencias. En la Fig.1.5 se indica la cantidad de trabajos de los que puede hacerse cargo una persona, mediante números encerrados en un recuadro. Lo que se hace en el modelo es asignar a las aristas que parten de la fuente hacia esos trabajos un valor igual al número de trabajos que puede atender.

Figura 1.5: Una ampliación del problema de asingnaciones



Nuestro modelo de asignaciones tiene la característica de que (el recurso p_i) que está antes escoge mejor (y tanto como desee) que el que está después, siempre y cuando no se afecte la optimabilidad de la asignación global. Esta propiedad le da a nuestras redes de preferencias un gran poder de modelación, capaces de representar problemas aplicados muy variados. En efecto, recordemos que en todo proceso de escogimiento, siempre se establece algún orden, por antigüedad, por edad, por altura, por tiempo de arribo, etc, este ordenamiento nos da la clave para ordenar los recursos. Por supuesto pueden haber situaciones en donde se desee una asignación arbitraria, pero esto es más la excepción que la regla; en todo caso siempre es posible conseguir algún método para barajar los recursos y resolver esta situación.

Descripción del software $\mathcal{H}$orario

Usando el modelo de asignaciones presentado anteriormente se ha desarrollado el software $\mathcal{H}$ orario que es capaz de construir un horario para un centro educativo. Aquí se explica en forma resumida su implementación y funcionamiento. Como se ha indicado antes, en este caso los recursos del problema lo constituyen los profesores conjuntamente con las materias que deben impartir y las disponibilidades de tiempo que tienen para cumplir con sus clases. Las necesidades están determinadas por los grupos que deben recibir cierta cantidad de lecciones con algunos de estos profesores. Veamos primero qué información debe alimentar el sistema. Para cada profesor es necesario saber cuáles días y en qué lecciones está dispuesto a trabajar. Además para cada una de las materias que imparte se debe indicar el nivel en que las impartes, los grupos o secciones que tiene a su cargo, el número semanal de lecciones que requiere cada grupo y (opcionalmente) alguna observación importante que debe tenerse en cuenta al armar el horario. Para cada grupo o sección se requiere saber en qué días y cuántas lecciones recibe. Para resolver el problema manualmente se puede emplear una pizarra o matriz con fichas (punteros). Cada fila representa una sección o grupo. Las columnas se usan para indicar los días y lecciones laboradas. En la Fig. 1.6 se muestra parte de esta pizarra que llamaremos P. En la posición $P[i, j]$ se coloca el nombre del profesor y la materia que imparte.

	Lunes-1	Lunes-2	$\cdots$	Viernes-8
1-1				
1-2				
$\vdots$				
5-4				

Figura 1.6: Tabla para horarios

Suponga que tenemos m profesores y consideremos el profesor p con $1 \leq p \leq m$. Supongamos que este docente debe impartir m materias distintas, entendiendo por "distintas" que dichas materias tienen diferente nombre o bien diferente nivel. Por limitaciones del modelo empleado, el algoritmo de repartición propuesto funciona en iteraciones, asignando primero la primera materia de cada profesor, luego la segunda y así hasta cubrir todas las materias de todos los profesores. En cada iteración del algoritmo de repartición, cada materia del profesor p representa entonces un recurso. Las necesidades vienen a estar formadas por las entradas de la matriz P presentada en la Fig 1.6. De esta forma se entenderá que una necesidad ha sido satisfecha cuando se logra colocar el nombre de un profesor, junto con una de las materias que imparte, en algunas de las entradas de la matriz P.

Como probablemente el lector habrá adivinado estamos tratando de armar el problema de manera que pueda resolverse usando redes de transporte y algoritmo de Ford-Fulkerson. Cuando se desea repartir la primera materia de cada profesor se debe crear una matriz de preferencias A. Si el profesor p puede trabajar el día d en la lección l, hacemos $A[p,k]=1$, en donde k es la casilla de P correspondiente al día d, lección l y sección s. Expliquemos un poco más esto. Las entradas de la matriz P pueden organizarse como si se tratasen de un vector, colocando, por ejemplo una fila después de la otra, o bien una columna después de la otra. La segunda opción no es tan recomendable debido a que esto haría que las lecciones semanales fuesen asignadas casi todas el mismo día, y esto, por razones pedagógicas, resulta inapropiado. Es mejor la primera opción. Por supuesto hay otras forma de organizar o bien indexar la matriz P (para tratarla como si fuese un vector) y esto dependerá del tipo de distribución que se desee. En $\mathcal{H}$orario, por ejemplo, se buscó que la repartición quedará gradeada, es decir, que de la sección o grupo 1-1, un profesor pasará al 1-2, después al 1-3, etc. Un esquema gradeado ayuda al docente en el sentido de que puede aplicar un mismo examen en varios grupos toda vez que no haya un receso de por medio. Además algunos estudiantes y el profesores prefieren un contacto de una lección pues les resulta menos cansado. Una vez que se ha creado la red de preferencias, el método de Ford-Fulkerson se encargará de hacer la mayor cantidad de asignaciones posibles. En este contexto debemos notar que si el flujo entre un recurso y una necesidad es de 1, significa que ha producido una asignación entre ambas.

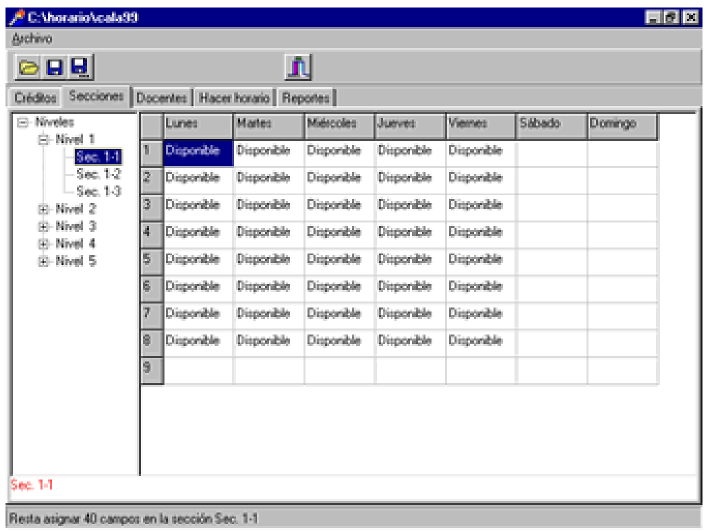


Figura1.7: Pantallas de Inicio y Creación de secciones.

El modelo de preferencias planteado tiene un defecto. Puede suceder que al profesor se le asigne en forma múltiple en una misma lección. Para corregir esto, en *H*orario se verifica que el profesor no esté ya asignado antes de efectuar cada asignación. Por supuesto que esta limitante hace que decrezca la eficiencia del algoritmo. No obstante, en la práctica *H*orario dio resultados entre aceptables y buenos. Dado que resulta difícil automatizar absolutamente todo el proceso, en *H*orario se incluye la posibilidad de realizar una asignación manualmente. Al dar click derecho en una casilla de la pizarra de asignaciones, el usuario puede saber cuáles profesores pueden ser asignados ahí, cuántas materias llevan asignadas, etc. Además de esto en *H*orario se pueden hacer asignaciones directamente sobre el horario de cada profesor brindando así un tratamiento muy puntual. Una vez que todas las actualizaciones manuales han sido llevadas a cabo, el usuario puede solicitar la asignación automática y *H*orario se encargará del resto. Aún más, una vez concluida dicha asignación, usando el modo manual, el usuario puede "retocar" el resultado generado. Hasta aquí se ha explicado en términos generales el algoritmo de repartición usado por el software *H*orario. A continuación se describe rápidamente dicho programa. En la Fig 1.7 se muestran las pantallas inicial (créditos) y la de creación de grupos o secciones.

Cuando se va introducir la información sobre un centro educativo, debe empezarse por indicar cuántos niveles tiene, cuántos grupos o secciones por nivel, qué días se labora, el número máximo de lecciones laboradas y el número típico de lecciones laboradas. *H*orario crea entonces un árbol con toda esta información. En las hojas de este árbol se despliega el horario de cada sección. En esta parte el usuario puede hacer modificaciones más puntuales sobre la disponibilidad de horas en las que se puede recibir lecciones. En la Fig.1.8 se muestran las pantallas correspondientes a la captura de la información de los profesores y la pizarra o matriz que almacena el horario en proceso. Sobre cada profesor se debe indicar en cuáles días y horas puede atender su carga académica. Una vez hecho esto, en una tabla de 10 filas (máxima cantidad de materias que puede impartir), se debe indicar el nombre de la materia, nivel, secciones que debe atender, número de lecciones semanales, observaciones importantes sobre esa materia, etc. Este módulo permite detectar situaciones extrañas o bien inconsistencias, como por ejemplo, que una misma sección sea atendida por más de un profesor, que en un grupo, una materia tenga más lecciones que en otra, siendo esta del mismo nivel. Se minimiza la cantidad de digitación al permitir al usuario escoger los datos a partir de algunos menúes.

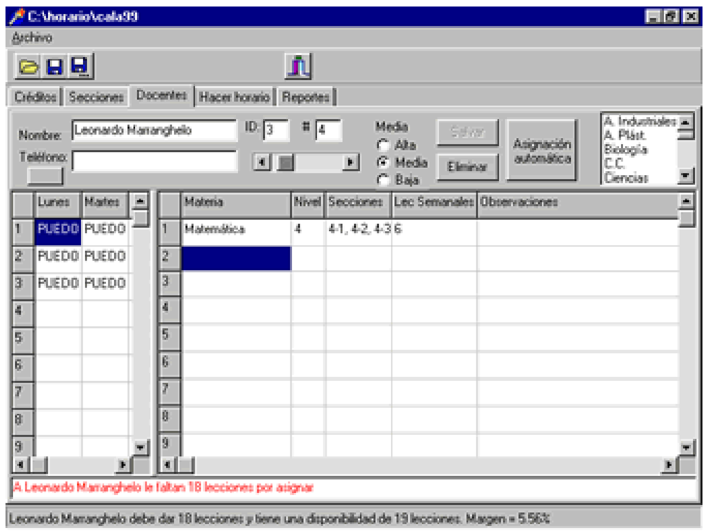
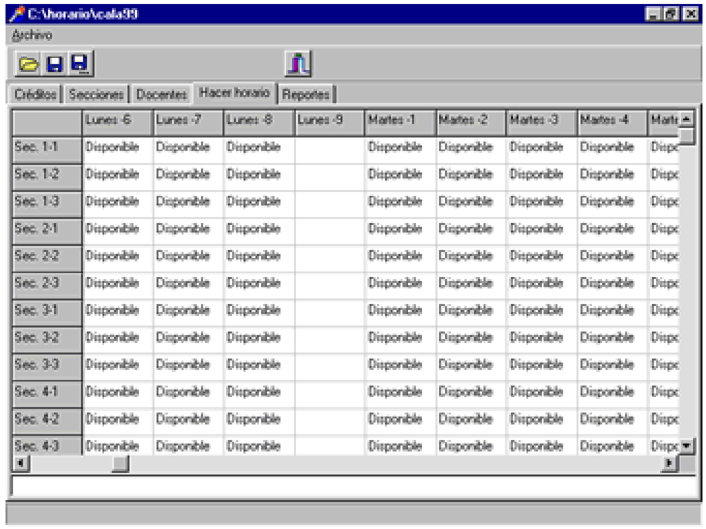


Figura 1.8: Información de los profesores y pizarra pra almacenar el horario.

En el módulo de captura de información del profesor se puede también realizar asignación de horario tanto manual como automáticamente. Esto le permite al usuario focalizar la atención en algún caso especial. La matriz o pizarra mostrada en la Fig.1.8 será para muchos las versión informatizada de una pizarra de madera que se emplea para registrar la construcción de un horario. La matriz luce como una hoja tipo MS-Excel con la ventaja de que con solo dar click derecho en una casilla tiene toda la información necesaria para realizar una asignación, por ejemplo cuántas lecciones de cada materia se han asignado en esa sección, qué profesores podrían ser asignados ahí, efectuar el mantenimiento de la pizarra (borrado de filas, columnas, reinicio), solicitar una asignación automática, averiguar qué profesores se encuentran a esa hora con alguna materia asignada, etc. Es importante aclarar que cada modificación que se realiza en cualquiera de los módulos se refleja en los demás de manera inmediata. Es decir el usario no está obligado a hacer múltiples modificaciones.



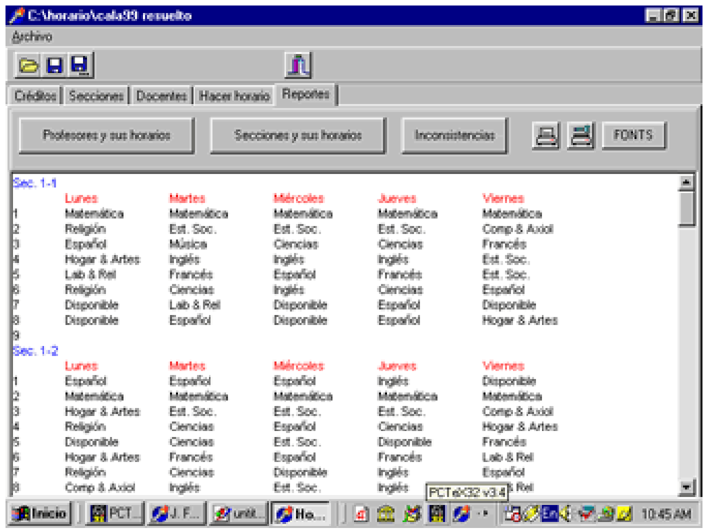


Figura 1.9: Reportes: horarios por profesor, por grupo. Inconsistencias

En la Fig.1.9 se muestran la pantalla correspondiente a los reportes disponibles. $\mathcal{H}$orario puede brindar tres tipos de reportes:

- un listado en que se presenta para cada profesor el horario confeccionado hasta ese momento,
- un listado en que se presenta para cada sección o grupo, las materias asignadas hasta ese momento.
- un listado en que se presentan las inconsistencias o situaciones extraordinarias detectadas.

Cada uno de los primeros dos reportes queda grabado en el Clipboard de Windows y puede ser pegado en MS-Word o MS-Excel para una posterior edición.

Conclusión

$\mathcal{H}$orario es una muestra clara de que la teoría de redes de transporte permite resolver problemas de la vida real. El algoritmo usado en modelo propuesto no es óptimo pero puede hacer una repartición de horarios de manera automatizada. Es importante hacer notar que $\mathcal{H}$orario puede hacer en minutos lo que a un humano le toma varios días o quizás semanas.

Bibliografía

1. A. V. AHO, J. E. HOPCROFT, J. D. ULLMAN, *Data Structures and Algorithms*, Addison-Wesley, Massachusetts, 1983.
2. M. S. BAZARAA, J. J. JARVIS AND H. SHERALI, *Linear Programming and Network Flows, 2nd Edition*, John Wiley & Sons, N.Y., 1990.
3. B. BOLLOBÁS, *Graph Theory*, Springer Verlag, NY, 1985.
4. G. BRASSARD, P. BRATLEY, *Algorithmics*, Prentice-Hall, N. J. USA., 1987.
5. T. CORMEN , C. LEISERSON Y R. RIVEST, *Introduction to Algorithms*, The MIT Press, Ms, USA, 1990.
6. P. E. GILL, W. MURRAY, M. A. SAUNDERS AND M. H. WRIGHT, *Mantaining LU Factors of a General Sparse Matrix*, Department of Operations Research, Stanford University, California, 1986.
7. F. S. HILLIER Y G. J. LIEBERMAN, *Introducción a la Investigación de Operaciones*, McGraw Hill, México, 1991.
8. R. JOHNSONBAUGH, *Matemáticas Discretas*, Grupo Editorial Iberoamérica, México, 1988.
9. M. SPIEGEL, *Ecuaciones Diferenciales Aplicadas*, Prentice-Hall, Bogota, 1983.
10. A. M. TENENBAUM Y M. J. AUGENSTEIN, *Estructuras de Datos en Pascal*, Prentice-Hall, México, 1981.

Derechos Reservados.