Revista de Matemática: Teoría y Aplicaciones **4**(2): 75–84 (1997)

ASIGNACIÓN ÓPTIMA DE RECURSOS

Juan Félix Avila Herrera[1]

Resumen

Este trabajo trata sobre asignación óptima de recursos. El modelo propuesto está basado en teoría de Flujo de Redes y se dan indicaciones para una implementación eficiente del modelo.

Abstract

This work is on optimal assignation of resources. The proposed model is based on Net Flow Theory and here we give some indications for an efficient implementation of the model.

1. El problema de las asignaciones

Estudiamos aquí el problema de asignar recursos respetando preferencias, toda vez que estas preferencias no violenten la optimabilidad. El modelo propuesto está basado en una técnica de Investigación de Operaciones conocida como Redes de Transporte. El lector que desconozca este tema puede consultar las referencias [2] y [5].

El problema de las asiganciones óptimas permite modelar muchas situaciones reales de interés. Veamos un caso. Considere un grupo de profesores a los que se les debe asignar cursos. Cada profesor está capacitado para impartir ciertos cursos, pero probablemente no todos. La idea consiste en repartir los cursos de acuerdo con las preferencias que manifiesten los docentes, de manera que no exista doble asignación. El método que se propone permite repartir los cursos en forma óptima; no obstante esto no significa que todos tendran su primera opción. De hecho si todos los profesores escogen impartir el curso "Sistemas Operativos", por ejemplo, obviamente no podemos asignárselo a todos. La idea es entonces complacer tanto como sea posible.

Podemos representar este tipo de problemas 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 suerte que lado $e \in E$, es incidente en vértice de V_1 y en otro de V_2.

Supongamos que tenemos 4 personas p_1, p_2, p_3, p_4, y 5 trabajos t_1, t_2, t_3, t_4, t_5. En la Fig.1 se indica mediante una arista con peso 1, cuando la persona p_i está calificada para el trabajo t_j.

[1]Escuela de Informática, Universidad Nacional, Heredia, Costa Rica

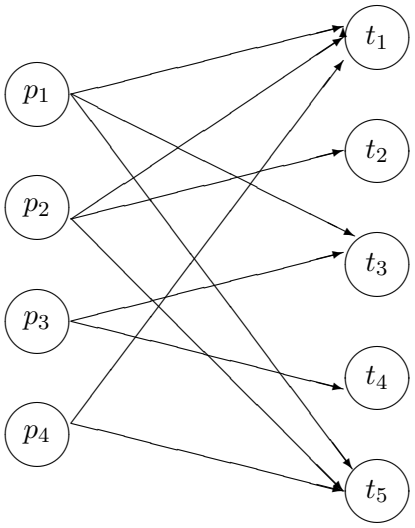

Figura 1: Representación de asignaciones mediante un grafo bipartido.

Para convertir el grafo bipartido de la Fig. 1 en una red de transporte es necesario agregar una super-fuente y un super-depósito. 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. El grafo resultante se aprecia en la Fig. 2.

Se debe observar que el peso asignado a cado lado en la red de la Fig. 2, es de 1. Estamos suponiendo aquí que cada persona p_i puede antender un único trabajo t_j, de los que manifiesta en la Fig. 1, 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. Explicaremos después en detalle esta variante.

Las siguientes definciones nos permiten formalizar aún más los conceptos que hemos venido estudiando:

Definición: 1 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 R a N es un conjuto $M \subseteq R \times N$.

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 super–fuente y el super–depósito,

   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,

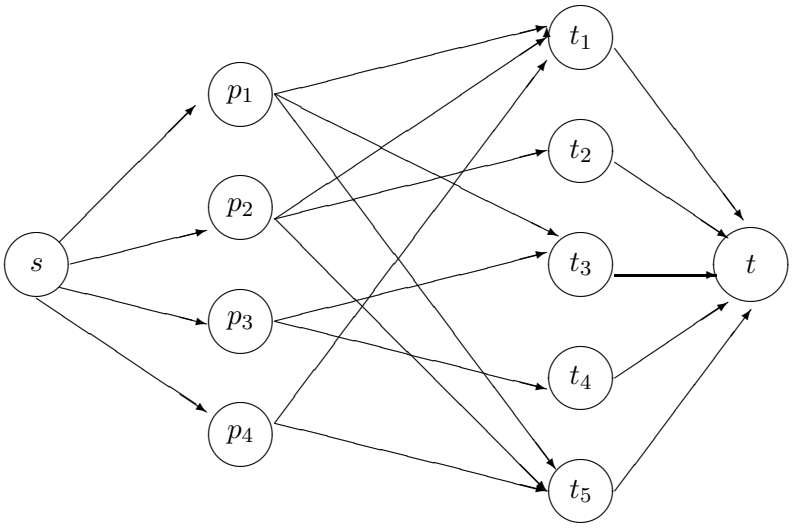

Figura 2: Red de transporte asignada al problema de las asignaciones.

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.

De esta forma para el problema expuesto en la Fig. 1, la matriz de capacidades de la red de preferencias es de tamaño 11×11, y se muestra en la Fig. 3.

$$cap = \left[\begin{array}{c|cccc|cccccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Bloque principal

Figura 3: Matriz de capacidades

El lector observador notará que la matriz de capacidades *cap* es rala, de hecho es suficiente con almacenar el bloque principal que se indica en la Fig. 3. En efecto, veamos

que los demás bloques filas y columnas son predecibles. La primera fila de esta matriz, indica las capacidades de las aristas que parten de la super–fuente s a los primeros m recursos (personas). Como no hay una arista de s a si misma, $cap[1,1] = 0$ siempre. Las siguientes $m = 4$ entradas de la primera fila valen 1, pues cada persona (recurso) puede tomar a lo sumo uno de los $n = 5$ trabajos (necesidades o labores). El resto de la primera fila es completada con ceros, pues no hay aristas entre la (super) fuente s y los trabajos o el destino t.

En el bloque principal (ver Fig. 3) se indican las preferencias. Observamos que por ejemplo las preferencias de la persona p_3 (trabajos 3, y 4) quedan plasmadas en las entradas

$$cap[1+3, 1+m+3] = cap[4,8] = 1, \quad \text{y} \quad cap[1+3, 1+m+4] = cap[4,9] = 1.$$

La columna $m+n+2 = 11$ es también predecible. De la posición 1 a la $m+1 = 5$ el valor es cero, puesto que no existe una arista entre la (super) fuente s o bien las personas p_i y el destino t. Desde la posición $m+2 = 6$ hasta la posición $m+n+2$, el valor es 1, puesto que t es un super–destino.

El resto de las entradas de la matriz *cap* tienen como valor 0, dejamos al lector como ejercicio verificar esto. Resumimos estos resultados en el siguiente teorema.

Teorema: **1** Sea *cap* la matriz de un problema de asignaciones con m recursos y n necesidades o tareas. Supóngase además que $\overline{cap}$ es una matriz de tamaño $m \times n$ que registra las preferencias de los recursos, de suerte que $\overline{cap}[i,j] = 1$ si el recurso i muestra preferencia sobre la necesidad j y 0 si no, entonces la matriz *cap* de la red de transporte asociada a este problema de asignaciones satisface:

$$cap[i,j] = \begin{cases} 1 & \text{si} \quad (i=1) \wedge (2 \le j \le m+1) \\ 1 & \text{si} \quad (j = m+n+2) \wedge (m+2 \le i \le m+n+1) \\ \overline{cap}[i-1, j-(n+1)] & \text{si} \quad (2 \le i \le m+1) \wedge (m+2 \le j \le m+n+1) \\ 0 & \text{en cualquier otro caso} \end{cases}$$

De acuerdo con el teorema 1 es suficiente con almacenar en memoria RAM la matriz $\overline{cap}$, los demás valores se pueden obtener mediante inferencias condicionales. De esta forma basta con mantener en memoria RAM tan solo el

$$\frac{100mn}{(m+n+2)^2} \tag{1}$$

por ciento de la matriz *cap*, que para el ejemplo de la matriz dada en la Fig. 3 es aproximadamente del 16.5 %. Esto significa entonces un ahorro 83.47 % para el caso mencionado. Notemos además que si $m = n$ (¡el peor caso! ¿Porqué?), la fórmula (1) se transforma en

$$\frac{100n^2}{(2n+2)^2}. \tag{2}$$

Si hacemos tender n a infinito en (2), obtenemos

$$\lim_{n\to+\infty} \frac{100n^2}{(2n+2)^2} = \lim_{n\to+\infty} \frac{100n^2}{4n^2} = \frac{100}{4} = 25.$$

De esta forma cuando hayan igual número de recursos e igual número de necesidades es suficiente representar tan solo el 25 % de la matriz *cap*. En la tabla presentada en la Fig. 4, se aprencia los distintos porcentajes que deben almacenarse de la matriz *cap* dependiento de su tamaño.

	2	4	8	16	32	64	128	256	512
2	11.111	12.500	11.111	8.000	4.938	2.768	1.469	0.757	0.385
4	12.500	16.000	16.327	13.223	8.864	5.224	2.851	1.492	0.763
8	11.111	16.327	19.753	18.935	14.512	9.350	5.377	2.894	1.503
16	8.000	13.223	18.935	22.145	20.480	15.229	9.608	5.456	2.916
32	4.938	8.864	14.512	20.480	23.508	21.324	15.607	9.741	5.496
64	2.768	5.224	9.350	15.229	21.324	24.237	21.766	15.802	9.808
128	1.469	2.851	5.377	9.608	15.607	21.766	24.614	21.993	15.900
256	0.757	1.492	2.894	5.456	9.741	15.802	21.993	24.806	22.107
512	0.385	0.763	1.503	2.916	5.496	9.808	15.900	22.107	24.903

Figura 4: Porcentaje que debe almacenarse en memoria RAM de la matriz *cap*

De esta forma se pone en envidencia que tomar en cuenta el resultado propuesto en el teorema 1, permite ahorrar al menos un 75 % de espacio en memoria RAM por concepto referente a la representación de la matriz *cap*.

Existe un aspecto que se ha dejado de lado en estas cavilaciones, y es el hecho de que se necesita de otra matriz (que llamamos $flow$ en el apéndice A) para mantener en memoria RAM el flujo actual a través de las aristas de la red. Sin embargo dado que por definición de flujo sobre una red de transporte, se debe satisfacer

$$cap[i, j] \leq flow[i, j],$$

las ideas aplicadas a *cap* pueden ser aplicadas $flow$. De hecho para la red de transporte representada en la Fig. 2, al aplicar el método Ford–Fulkerson obtenemos la matriz mostrada en Fig. 5.

La primera fila y la última columna de $flow$ son predecibles debido a la presencia de una super–fuente s y un super–depósito t. Los bloques que eran cero en la matriz *cap* permanecen igual en $flow$, es más, el bloque principal en $flow$ posee todavía más entradas nulas que *cap*. La justificación de esto es simple. Una vez aplicado el método de Ford–Fulkerson, se debe asignar a cada recurso (persona) a lo sumo una de sus solicitudes, esto significa que de la fila 2 a la fila $m + 1$ cada fila exhibirá a lo sumo un 1. De hecho si un recurso queda sin asignación, la fila correspondiente será completamente nula.

Basados en lo anterior es plausible pensar en una estructura de datos *ad hoc* para $flow$, sin embargo tal vez esto no sea lo más apropiado. La justificación del "tal vez" es la siguiente. Si simplemente se desea que cada persona tenga la posibilidad de quedarse con a lo sumo una de sus preferencias, la idea es buena, y por tanto sería conveniente que la estructura de datos diseñada para $flow$ tomara partido o ventaja de las características

$$flow = \left[\begin{array}{c|cccc|cccccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Bloque principal

Figura 5: Matriz final de flujos al aplicar Ford–Fulkerson

que se han citado antes. Sin embargo, es posible tal y como se mencionó en la página 76, que un recurso pueda hacerse cargo de más de una necesidad o tarea.

Veamos ahora en detalle una variante del problema que hemos venido desarrollando a lo largo de este trabajo, y que ya se ha mencionado antes. Supongamos que en una manifestación preferencial (como la dada en la Fig. 2), la personas p_1 y p_3 pueden cada uno hacerse cargo de dos de sus preferencias. En la Fig. 6 se indica la cantidad de trabajos de los que puede hacerse cargo una persona, mediante números encerrados en un rectángulo. 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.

Por supuesto que se puede pensar en casos más complicados en los que, por ejemplo la persona p_{24} puede atender simultaneamente 38 de las 40 preferencias que manifestó previamente. Sin embargo para fijar ideas, el ejemplo anterior es apropiado. En la Fig. 7 se aprecian conjutamente las respectivas matrices de capacidades *cap* y flujos $flow$.

Este caso nos brinda mucha información interesante. Observamos primero que en la fila 1 de *cap*, de la posición 2 a la posición $m+1$, los valores pueden ser diferentes de 0 y 1. En efecto, aquí se puede apreciar cuál es la capacidad de atención de cada recurso. Así por ejemplo, $cap[1,4] = 2$, indica que la persona $4-1=3$ puede hacerse cargo de 2 labores.

La última fila de $flow$ es también reveladora. Notamos que de la posición $m+2$ a la posición $m+n+1$ los valores continúan siendo binarios (0 ó 1). Recordemos que un flujo de una unidad enviado desde un trabajo t_i a la super–fuente t, indica simplemente que ese trabajo fue asignado a alguna persona p_j.

Observamos también en la matriz $flow$ algo interesante y que exhibe un aspecto clave de esta investigación. En la fila 2, los valores $flow[2, m+2] = flow[2, m+4] = 1$. Esto significa simplemente que las preferencias manifestadas por la persona p_1, solicitando los trabajos t_1 y t_3 fueron atendidas en forma preferencial; sin importar que la persona p_3 (que solicitó únicamente el trabajo t_3) quedara sin asignación (obsérvese que la fila $3+1$ de $flow$ es completamente nula).

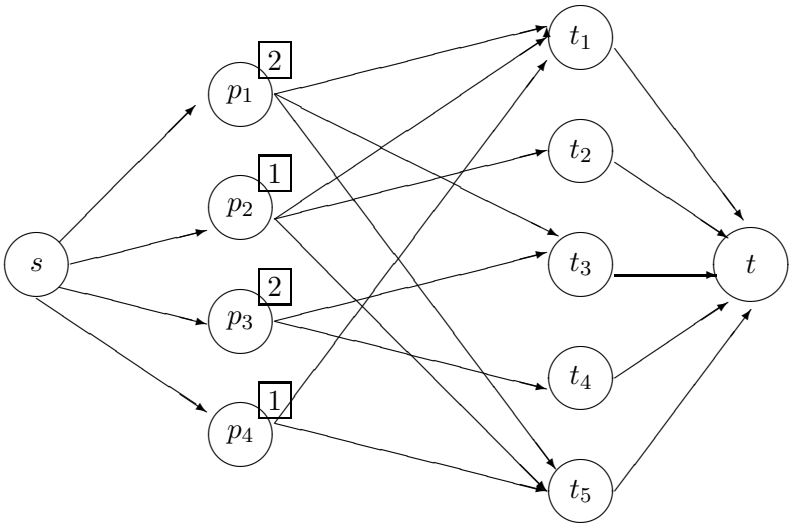

Figura 6: Una ampliación del problema de asingnaciones

En la anterior observación se ponde en evidencia una característica fundamental de este modelo: *la asignación preferencial respetando orden de solicitud, toda vez que no se sacrifique optimabilidad*. En efecto, la forma en que se ha modelado el problema de asignaciones, y la utilizanción del método de Ford–Fulkerson son los causantes de este efecto, por demás deseado. La justificación de este fenómeno estriba simplemente en la forma en que el algoritmo de Ford–Fulkerson escoge sus rutas aumentantes, y el hecho de que el problema de asignaciones se pueda representar mediante un grafo bipartido (que se transforma en una red de transporte mediante la adición de una super–fuente y un super–depósito). El algoritmo busca la primera ruta aumentante empezando con la superfuente s, y tomando como segundo eslabón el nodo etiquetado con p_1. Una vez que se ha arribado

$$cap = \left[\begin{array}{c|cccc|cccccc} 0 & 2 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \quad flow = \left[\begin{array}{c|cccc|cccccc} 0 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Figura 7: Matriz de capacidades y flujos

a p_1, se escoge la arista que modela la primera preferencia de la persona p_1 En el ejemplo de la Fig. 7, esto hace que el siguiente eslabón en la ruta aumentante sea t_1, y obviamente para complentar la ruta, el último eslabón es el super–depósito t.

Al buscar la siguiente ruta aumentante se sigue la misma estrategia, sin embargo la arista que une a la persona p_1 con su primera preferencia ya está saturada, por tanto se debe optar por la segunda preferencia, toda vez que ésta exista. En el caso de la Fig. 7, ésta existe y por lo tanto se produce la segunda asignación.

Una vez que la capacidad de la arista que sale de la super–fuente y llega a p_1 está agotada, se busca la siguiente ruta aumentante utilizando como segundo eslabón a p_2. La asignación de una preferencia a p_2 está supeditada a que esa escogencia no haya sido asignada previamente. Recordemos que dicha asignación quedó registrada en la matriz de flujos utilizando un 1 en la arista que sale de esa preferencia y va hacia el super–depósito t, imposibilitando a esta arista formar parte de una ruta aumentante.

La segunda forma de mejorar un flujo sobre una red (ver [10]), es la que puede dar al traste con el orden en el que se haya solicitado un recurso. En el método de Ford–Fulkerson, este segundo enfoque revisa las semivías existentes, determinando si es necesario disminuir el flujo sobre una tubería para aumentarlo en otra.

Resumimos las virtudes de nuestro modelo de asignaciones en el siguiente teorema.

TEOREMA: **2** Sean $P = \{ p_1, p_2, \cdots, p_m \}$ m recursos y $R = \{ t_1, t_2, \cdots, t_n \}$ n necesidades, M una manifestación preferencial de P en R y $G = (V, E)$ la red de preferencias asociada a M. Sea a un vector de m posiciones tal que $a[i]$ es el número de necesidades (de las manifestadas en M) que puede atender el recurso p_i. Si ordenamos los nodos de G como

$$V = \{ s \} \cup R \cup N \cup \{ t \},$$

de suerte que s es el primero, p_1, es el segundo, $\cdots$, p_m es el $(m+1)$–ésimo, t_1 es el $(m+2)$–ésimo, $\cdots$, t_n es el $(m+n+1)$–ésimo y t el $(m+n+2)$–ésimo, y además de esto, asignamos un peso de $a[i]$ $(1 \leq i \leq m)$ a la arista que sale de la super–fuente s y llega al recurso p_i, entonces al aplicar el método de Ford–Fulkerson a este modelo, éste respeta el orden, asignado todas las preferencias factibles a un nodo w, toda vez que algunas o todas no hayan sido previamente asignadas a un nodo precedente v, ni se violente la optimabilidad de la asignación global.

El teorema anterior simplemente dice que el 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 globlal. 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 esto solventaría una tal situación.

2. El programa "Matri.exe"

Como fruto de las ideas formuladas en este trabajo se desarrolló un programa bautizado "Matri.exe". Este programa es una versión para DOS capaz de efectuar parejamientos o asignaciones de hasta 175 recursos con hasta 175 necesidades (recordemos que no tienen que coincidir el número de recursos con el número de necesidades). El programa fue desarrollado por el autor, siguiendo de cerca las consideraciones formuladas en los anteriores secciones, de suerte que el resultado es una herramienta mucho más poderosa que la se obtendría con una implentación simple del método de Ford–Fulkerson sobre la red de transporte respectiva.

Un ejemplo de la entrada o *input* del programa en cuestión es un archivo tipo texto (con nombre "Ejemplo.ff") con la siguiente estructura

```
10
15
9,   14,  14,  < 1 >
4,   10,  12,  < 3 >
8,   3,   15,  < 3 >
15,  1,   4,   < 3 >
3,   5,   13,  < 1 >
9,   11,  14,  < 1 >
14,  2,   8,   < 3 >
13,  12,  10,  < 2 >
13,  2,   14,  < 3 >
8,   12,  12,  < 1 >
```

Las primeras dos cantidades indican los recursos (10) y las necesidades (15). Seguidamente se indican las preferencias de cada uno de esos recursos y el número de necesidades que es capaz de atender. Así por ejemplo el recurso # 8, muestra interés por las necesidades 13, 12 y 10; sin embargo es capaz de atender solo 2 de esas necesidades.

La salida del programa "matri.exe", es también un archivo tipo texto. Para el ejemplo anterior el *output* o salida del programa es el siguiente archivo (con nombre"Ejemplo.fff")

```
SOBRE PROBLEMA Ejemplo.ff
10 Número de elementos del dominio
15 Número de elementos del codominio

1 quiere:   9, 14,          y puede con [1] ⇒   < 9 >
2 quiere:   4, 10, 12,      y puede con [3] ⇒   < 4 >< 10 >< 12 >
3 quiere:   3, 8, 15,       y puede con [3] ⇒   < 3 >< 8 >< 15 >
4 quiere:   1, 4, 15,       y puede con [3] ⇒   < 1 >
5 quiere:   3, 5, 13,       y puede con [1] ⇒   < 5 >
6 quiere:   9, 11, 14,      y puede con [1] ⇒   < 11 >
7 quiere:   2, 8, 14,       y puede con [3] ⇒   < 2 >< 14 >
8 quiere:   10, 12, 13,     y puede con [2] ⇒   < 13 >
9 quiere:   2, 13, 14,      y puede con [3] ⇒   Quedó sin asignación !!
10 quiere:  8, 12,          y puede con [1] ⇒   Quedó sin asignación !!
```

Total de tareas asignadas : 13 Total de "Sin asignación": 2

Software diseñado por Juan Félix Avila Herrera, Julio 1995
Escuela de Informática, Universidad Nacional

De esta forma el recurso # 8 que manifestaba preferencias por las necesidades 10, 12 y 13 (y podía atender a lo sumo 2 de éstas) fue asignado solo a la necesidad 13. El programa "matri.exe" hizo 13 asignaciones y dejó 2 recursos sin asignación. Es importante enfatizar que este es el máximo de asignaciones que se puede hacer, sin embargo pueden existir distintas formas de lograr este máximo.

La forma de utilizar el programa es (en el ejemplo que se ha venido explicando)

Matri Ejemplo.ff

que se hace directamente desde la línea de comando del DOS. De esta forma es posible hacer la modelación de cualquier problema utilizando cualquier lenguaje (Pascal, Cobol, Fox, C^{++}, etc); se genera un archivo de entrada (como "Ejemplo.ff"), se hace una llamada a "matri.exe" y se interpreta el archivo de salida (como "Ejemplo.fff"). De esta forma "matri.exe" es un software que se encarga de hacer el "trabajo sucio". Lo referente a interfaz, modelación e interpretación queda a cargo del analista que opta por "matri.exe" para resolver su problema. Esto es una ventaja pues da mucha libertad para resolver problemas con muy distinta apariencia, pero con un mismo transfondo.

Referencias

[1] Aho, A.V.; Hopcroft, J.E.; Ullman, J.D. (1983) *Data Structures and Algorithms*. Addison–Wesley, Reading Mass.

[2] Bazaraa, M.S.; Jarvis, J.J.; Sherali, H. (1990) *Linear Programming and Network Flows*, 2nd Edition. John Wiley & Sons, New York.

[3] Bollobás, B. (1985) *Graph Theory*. Springer-Verlag, New York.

[4] Brassard, G.; Bratley, P. (1987) *Algorithmics*. Prentice–Hall, Englewood-Cliffs, N. J.

[5] Cormen, T.; Leiserson, C.; Rivest, R. (1990) *Introduction to Algorithms*. The MIT Press, Cambridge Mass.

[6] Gill, P.E.; Murray, W.; Saunders, M.A.; Wright, M.H. (1986) *Mantaining LU Factors of a General Sparse Matrix*. Department of Operations Research, Stanford University, Calif.

[7] Hillier, F.S.; Lieberman. G.J. (1991) *Introducción a la Investigación de Operaciones*. McGraw Hill, México.

[8] Johnsonbaugh, R. (1988) *Matemáticas Discretas*. Grupo Editorial Iberoamérica, México.

[9] Spiegel, M. (1983) *Ecuaciones Diferenciales Aplicadas*. Prentice–Hall, Bogotá.

[10] Tenenbaum, A.M.; Augenstein, M.J. (1981) *Estructuras de Datos en Pascal*. Prentice–Hall, México.