Estrategias Óptimas de Evacuación

Máster Universitario en Matemática Avanzada

Trabajo Fin de Máster

Autor:
Miguel Marín López
Tutor:
Alfredo Marín Pérez

UNIVERSIDAD DE MURCIA

Junio 2020

Estrategias Óptimas de Evacuación

Flujos dinámicos y aplicación a un caso práctico

Autor
Miguel Marín López

Tutor
Alfredo Marín Pérez
Departamento de Estadística e Investigación Operativa

Facultad de Matemáticas
Máster Universitario en Matemática Avanzada

FACULTAD DE MATEMÁTICAS

UNIVERSIDAD DE MURCIA

MURCIA, Junio 2020

Declaración de originalidad

Miguel Marín López, autor del Trabajo de Fin de Máster *Estrategias Óptimas de Evacuación*, bajo la tutela del profesor Alfredo Marín Pérez,

DECLARA

que el trabajo que presenta es original, en el sentido de que ha puesto el mayor empeño en citar debidamente todas las fuentes utilizadas.

En Murcia, a 22 de junio de 2020

Fdo.: Miguel Marín López

Nota: En la Secretaría de la Facultad de Matemáticas se ha depositado una copia firmada de esta declaración.

A Maria Massone,
por su apoyo incondicional.

Sólo podemos ver un poco del futuro,
pero lo suficiente para saber
que hay mucho por hacer.

Alan Turing.

Agradecimientos

En primer lugar quisiera agradecerle a mi tutor, D. Alfredo Marín Pérez, por su apoyo y dedicación para con este trabajo. Su guía sobre en qué dirección avanzar ha facilitado que este trabajo tenga el rigor que merece. Gracias por aceptar llevarlo y la confianza que me ha transmitido.

Asimismo, agradezco al Equipo Directivo del IES Francisco Salzillo por su amabilidad y diligencia, dándome acceso a los datos e informes necesarios para llevar a cabo el caso práctico que se expone en el último capítulo.

No quisiera perder la oportunidad para agradecerle a mi familia y mi entorno más cercano, su apoyo este año ha resultado fundamental para la consecución final de este trabajo.

A todos, muchas gracias.

Índice general

Resumen

Cuando se trata de evacuar una zona o lugar, lo que debe primar por encima de todo es el **bienestar** de las personas implicadas. La planificación previa a una evacuación es un problema en el que las matemáticas son fundamentales para dejar el menor número de variables al puro azar. Encontrar rutas de escape óptimas por flujo de personas (Problema del flujo máximo dinámico) o por el tiempo que se tarda en recorrerlas (Problema del flujo más rápido), e inclusive, óptimas por el flujo en cada instante de tiempo (Problema del flujo de llegada más temprana); todas estas cuestiones no son triviales, dependen del entorno, pudiendo cambiar las condiciones a lo largo de la propia evacuación.

Nosotros nos aproximaremos a estos problemas desde la Teoría de Grafos. En el **Capítulo 1** de este trabajo daremos una introducción a dicha teoría, fijando el objeto con el que modelizaremos las situaciones de evacuación: **las redes** (ver Definición 1.1.2). Por ejemplo, un edificio, una red de carreteras o una ciudad, son susceptibles de ser representados por una red. Los nodos representarán las diferentes habitaciones, ciudades o cruces de calles; mientras que los arcos serán las conexiones entre dichos nodos.

Durante una evacuación, las personas se moverán por los arcos y este movimiento será modelizado mediante una función de **flujo estático** (ver Definición 1.2.1). Teniendo en cuenta las cualidades físicas de aquello que representan los arcos, se definirán funciones de capacidad, $c : A \longrightarrow \mathbb{R}^+$, y de tiempo de tránsito por ellos, $\tau : A \longrightarrow \mathbb{R}$. Además, se distinguirán un par de nodos $s, t \in V$, que serán la fuente y el destino, respectivamente, de dicho flujo (ver Figura 1.2).

Seguidamente, en la Sección 1.2, plantearemos la versión estática del problema del **flujo máximo** en redes, ver Formulación (1.2), en la que se optimiza el valor de un flujo estático, que es la cantidad que llega al destino t y que, por conservación del mismo, es la cantidad que sale desde la fuente s (ver Ecuación (1.1)).

Se describirán los Algoritmos 1 y 2, que debemos a Ford Jr y Fulkerson (1956, 1962), y resuelven de manera exacta el problema del flujo máximo estático en una red mediante el uso de la red residual (ver Definición 1.2.7). Además, se verá la estrecha relación del flujo de una red con el concepto de **corte en una red** (ver Definición 1.2.2). Culminando en el Teorema del flujo máximo - corte mínimo, el cual utilizaremos en varias ocasiones para demostrar que un flujo obtenido es máximo o no lo es (ver Ejemplos 2.2.4 y 2.2.8).

A continuación, en la Sección 1.3, se usará la función de tiempo de tránsito τ como coste de atravesar un arco, así se considerará el problema de encontrar un **flujo a coste mínimo** con la Formulación (1.3). Se presentará el Algoritmo 3, que se debe a Busacker y Gowen (1961), y que servirá de apoyo fundamental para los algoritmos de flujos dinámicos.

Terminaremos tratando la localización de instalaciones en una red por la que pasa un flujo en la Sección 1.4. Su estudio nos permitirá plantear el Problema q-FlowLoc para saber cuáles son las posiciones óptimas en las que se deben colocar esos elementos en nuestra red, afectando lo menos posible al flujo, en nuestro caso, de una evacuación. En la Definición 1.4.1 se construye una red especial, llamada **red de flujo y localización**, que será utilizada como objeto matemático sobre el que resolver dicho problema. Se planteará su programación lineal entera en la Formulación (1.4) y se demostrará, en el Teorema 1.4.3, que es un problema $\mathcal{NP}$-completo (ver Anexo A.1 para ampliar).

Esto motivará la construcción de **heurísticas** que den buenas soluciones al problema en tiempo polinomial. Nosotros presentaremos el Algoritmo 5, cuyo desarrollo le debemos a Hamacher et al. (2013). Este se basa en colocar, por orden de tamaño, las instalaciones en los arcos de la red en los que, individualmente, se afecte lo menos posible al flujo máximo e ir recalculándolo para cada instalación colocada.

En el primer capítulo nos centraremos en los llamados flujos estáticos y los problemas que nacen a partir de esa noción. Sin embargo, en las evacuaciones, la componente temporal, que es lo que se tarda en salir de una zona o transitar unos arcos, es indispensable. En el **Capítulo 2** empezaremos fijando un horizonte temporal H sobre el que trabajar, y se construirá el concepto de **red expandida temporalmente**, en la Definición 2.1.2, que nos ayudará a replantear, cuando sea necesario, los problemas que se den en el capítulo en términos de flujos estáticos.

Al igual que en el caso estático, en la Definición 2.1.3, se crea la noción de **flujo dinámico**, que recogerá el efecto de la función de coste de tránsito τ en el flujo y el horizonte temporal. Además, de forma análoga, se definirá el valor de un flujo dinámico como la cantidad que llega al nodo destino t en el horizonte temporal H y que, por conservación del mismo, será igual a la cantidad que sale del nodo fuente s hasta el horizonte H (ver Ecuación (2.1)).

En la Sección 2.2 se planteará formalmente el Problema del flujo máximo dinámico para un horizonte temporal H. Veremos, gracias a Ford Jr y Fulkerson (1958), que no es necesario utilizar el concepto de red expandida temporalmente para resolver este problema y que siempre tiene una solución óptima en una subclase de flujos dinámicos, los **flujos repetidos temporalmente** (ver Definición 2.2.3).Se llega a una expresión explícita para el valor de un flujo en esta subclase con el Teorema 2.2.7. Usando el Algoritmo 6, será suficiente encontrar flujos máximos estáticos a coste mínimo en la red para resolver el problema del flujo máximo dinámico.

Después se hablará del **corte dinámico** en una red (ver Definición 2.2.10. Este se verá como una generalización de un corte en el caso estático y se concluirá, por medio de teoría de dualidad de programación lineal, que la capacidad de un corte dinámico mínimo es igual al valor de un flujo máximo dinámico.

A continuación, en la Sección 2.3, se presentarán en la Definición 2.3.1 los flujos de llegada y salida en una red y, con ellos, se plantearán los **Problemas del flujo de llegada/salida más temprana/tardía**. Estos son diferentes a un flujo máximo dinámico convencional, en el sentido de que son máximos para cada instante de tiempo. Poseen una gran versatilidad en la planificación, no solo de evacuaciones, sino de múltiples actividades; como por ejemplo, si el almacenamiento de un producto es muy caro frente a su coste de transporte, se preferirá que llegue lo más tarde posible al destino.

Con la Proposición 2.3.3 probaremos que siempre existe un flujo de llegada más temprana y, con el Corolario 2.3.4, el resto de flujos también existen siempre en una red. El Algoritmo 7, que le debemos a Minieka (1973), nos permite construir explícitamente, en tiempo *pseudo-polinomial* (ver Anexo A.1), un flujo de llegada más temprana usando un cálculo sucesivo de flujos a coste mínimo. Después, nos apoyaremos en el Teorema 2.3.7 para ver que siempre existen flujos que combinan las características de llegada más temprana/tardía con la salida más temprana/tardía. A cualquier flujo dinámico que sea solución de dos de los problemas mencionados se le llamará **flujo máximo universal** (ver Definición 2.3.9) y veremos, en la Proposición 2.3.6, que el flujo obtenido en el Algoritmo 7 es un flujo máximo universal.

El la Sección 2.4 resolveremos el último de los problemas que hemos mencionado, el **Problema del flujo más rápido**, que trata de minimizar el horizonte temporal H mientras se mantiene un valor de flujo $v_H(x)$ mayor que una cota dada K. Se verá, con el Lema 2.4.2, que existe una caracterización, en términos del flujo máximo dinámico, muy intuitiva. Este demuestra que si un flujo máximo, en tiempo $H-1$, no llega a la cota K, entonces cualquier flujo factible en tiempo H que llegue a la cota K es un flujo más rápido. En el trabajo de Burkard et al. (1993) se presentan varios algoritmos que aprovechan las propiedades de la función $f(H) := \max_x v_H(x)$ para encontrar, de manera eficiente, un flujo más rápido. Aquí desarrollaremos dos, el Algoritmo 8, que se basa en una búsqueda binaria optimizada; y el Algoritmo 9, que usa el método de Newton para encontrar un cero de la función $K - f(H)$.

Como colofón de esta sección se demostrará el **Teorema de triple optimización**, debido a Jarvis y Ratliff (1982). Este nos permite asegurar que, si un flujo es solución del problema de llegada más temprana, entonces también es solución del problema del flujo más rápido. Además, también sería un flujo mínimo ponderado de llegada, es decir, dicho flujo minimiza el tiempo medio que se tarda en llegar hasta t.

Se planteará en la Sección 2.5, de forma análoga al caso estático, el problema de colocar q instalaciones en los arcos de una red, por la que se define un flujo dinámico, de manera que afecte lo menos posible al valor del flujo. Si interpretamos este problema con la red expandida temporalmente, colocar una instalación en un arco es equivalente a colocarla en todas sus copias temporales, como se puede ver en la Figura 2.18. Por el Teorema 2.2.7 sabemos que un flujo máximo dinámico puede ser calculado a través de un flujo estático a coste mínimo. Esto nos permitirá formularlo como problema de programación entera, ver la Formulación (2.4). Como sabemos que el problema es $\mathcal{NP}$-completo, se proporcionará una heurística en tiempo polinomial en el Algoritmo 11, cuyo desarrollo le debemos a Hamacher et al. (2013).

En el **Capítulo 3** se aplicarán las técnicas descritas a la resolución de un **caso real** de evacuación en un instituto de la Región de Murcia. Al principio del capítulo se introducirá una técnica utilizada en evacuaciones reales, el **contraflow**; que se basa en invertir, de manera inteligente, el sentido de determinados arcos de una red. Veremos que el problema de encontrar el conjunto óptimo de arcos a invertir es $\mathcal{NP}$-completo (ver Teorema 3.1.1).

Después, en la Sección 3.2, se construirá la red que represente al instituto (ver Figuras 3.5, 3.6, 3.7 y 3.8) y se resolverá el problema del flujo máximo dinámico y el flujo más rápido, usando la red expandida temporalmente, mediante los Códigos A.1 y A.2. En la Sección 3.3, se resolverá el problema del q-FlowLoc, para $q = 10$, aplicado al instituto mediante una modificación del Algoritmo 5, que se puede ver en el Código A.3, para poder tratar con redes expandidas temporalmente.

Palabras clave: Evacuación, redes, flujos dinámicos, q-FlowLoc.

Miguel Marín López
22 de junio de 2020

1. Preliminares

En este capítulo fijaremos la notación que utilizaremos de aquí en adelante definiendo los conceptos de grafo, tanto dirigido como no dirigido, y el de red dirigida ya que será este último el instrumento central que usaremos en la teoría que se desarrollará posteriormente.

Seguidamente pasaremos a definir lo que entenderemos como *flujo factible estático* sobre una red dada, lo que consideraremos un *corte de la red* y la relación íntima que guardan culminando en el Teorema 1.2.11 de flujo máximo-corte mínimo. En el resto de secciones nos ocuparemos de otros problemas tipo con flujos estáticos.

Formularemos el problema del flujo estático máximo en una red como problema de programación lineal, aunque también daremos unos algoritmos (Algoritmo 1 y 2) que permitirán hallar dicho flujo máximo de manera constructiva. Además, dada su necesidad posterior, introduciremos el problema del flujo estático a coste mínimo y, como en el caso del flujo máximo, daremos una formulación lineal y un algoritmo (Algoritmo 3) para resolverlo.

A continuación veremos los problemas de flujo y localización, que denotaremos como *Flow-Loc*, en redes y cuya aplicación directa se puede encontrar en los planes de evacuación de edificios o ciudades. Asumiremos que de aquellos objetos que queremos localizar, llamados instalaciones, en cada arco, el más grande de todos ellos afectará al flujo aunque podremos considerar el caso de que las instalaciones en un mismo arco resten capacidad según la suma de sus tamaños.

Daremos una formulación lineal entera mixta del problema FlowLoc en el que queramos maximizar el flujo que quede tras la localización. Demostraremos que este problema es $\mathcal{NP}$-completo y daremos un algoritmo exacto (Algoritmo 4) para poder encontrar soluciones óptimas en el caso de querer localizar una sola instalación y un algoritmo heurístico en el caso de localizar varias (Algoritmo 5).

Las referencias principales para este capítulo son Ford Jr y Fulkerson (1962); Pelegrín et al. (1992); Ahuja et al. (1993), en las que se puede encontrar toda la teoría básica desarrollada aquí y más conceptos que no utilizamos en este trabajo. Para el modelo de FlowLoc nos basaremos en el artículo de Hamacher et al. (2013).

1.1. Grafos y Redes

Un grafo es una forma de representación de las relaciones que tienen un conjunto de objetos respecto a una característica. Ese conjunto de objetos, llamados *nodos*, se representará con la letra V. Las relaciones que tengan estos nodos será un subconjunto $X \subseteq V \times V$. Entonces diremos que i está relacionado con j para $i, j \in V$ si $(i, j) \in X$.

Definición 1.1.1. Un *grafo* es un par $G = (V, X)$ donde V es el conjunto de nodos y $X \subseteq V \times V$.

Diremos que un grafo es *no dirigido* si las relaciones del conjunto X son simétricas, es decir, $(i, j) \in X \Leftrightarrow (j, i) \in X$. En ese caso denotaremos al conjunto $E := X$ y llamaremos a sus elementos *aristas*.
Sin embargo, si la relación no es simétrica, diremos que los elementos de X son pares ordenados y entonces denotaremos por $A := X$, llamando a sus elementos *arcos* y definiendo el par $G = (V, A)$ como *grafo dirigido*.

Además, se dirá que un *grafo es simple* si no tiene aristas/arcos $(i, i) \in A$ con $i \in V$ un nodo, es decir, si no posee lo que se denominan bucles. Y diremos que un grafo es un *multigrafo* si entre algún par de vértices $i, j \in V$ existe más de un arco $(i, j) \in A$.

Figura 1.1: Ejemplo de grafo con bucle (izquierda) y multigrafo (derecha)

A partir de ahora trabajaremos sobre grafos dirigidos pues nos permiten representar una mayor cantidad de situaciones en la vida real que no tengan simetría (sentido de las carreteras, redes eléctricas, etc). Una vez definidas las relaciones entre los nodos podemos definir a su vez funciones sobre estas relaciones, es decir, sobre los arcos, que nos permitirán modelizar una gran cantidad de problemas, entre ellos de localización (almacenes, escuelas, vertederos, bomberos, etc) y de flujos en la red (servicios de aguas, evacuaciones, paquetería, etc). Para una lista más completa de aplicaciones se recomienda Ahuja et al. (1993).

También supondremos en este documento que estos grafos son simples y *no* son multigrafos a no ser que se especifique lo contrario. Si bien es cierto que esta asunción quita generalidad al modelo, para el trabajo que vamos a desarrollar no serán necesarios y causarían más confusión por una notación más farragosa.

Definición 1.1.2. Una *red dirigida* es un par $R = (G, F)$ donde G es un grafo dirigido y F un conjunto de funciones que están definidas en A y cuyas imágenes están en $\mathbb{R}$.

Ejemplo 1.1.3. La 5-upla $((V, A), (f, c, \tau))$ es una red con $c : A \to \mathbb{R}^+$ una función que asigna una capacidad máxima (*ceiling*) a un arco, $f : A \to \mathbb{R}^+$ la capacidad mínima (*floor*) de cada arco y $\tau : A \to \mathbb{R}$ el coste/tiempo de usar un arco.

Denotaremos por $\Gamma(i) := \{(i, j) \in A\}$ al conjunto de arcos *sucesores* del nodo i, diremos que estos arcos *salen* del nodo i, y por $\Gamma^{-1}(i) := \{(j, i) \in A\}$ al conjunto de arcos *predecesores* del nodo i, diremos que estos arcos *entran* al nodo i. Dada una red dirigida distinguiremos dos nodos $s, t \in V$ que llamaremos nodo fuente y nodo destino de nuestra red y podremos, sin pérdida de generalidad (véase Figura 1.2), suponer que $\Gamma(t) = \Gamma^{-1}(s) = \emptyset$, es decir, no existen arcos en la red que entren a s y no hay arcos en la red que salgan de t.

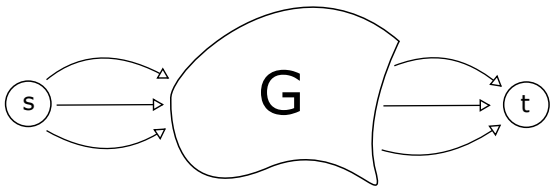

Figura 1.2: Si en el grafo G no existen los nodos s y t, siempre podemos añadirlos

También existen modelos de redes donde hay varios nodos fuente y varios destino. Esto sirve para modelizar, por ejemplo, demandas distintas que se deben cubrir en los diferentes destinos mediante diversos focos de oferta, un edificio con múltiples salidas y que todas ellas son distinguibles por alguna razón, etc. Nosotros nos restringiremos al caso de una sola fuente y un solo destino acumulando todos los nodos fuente en uno y todos los nodos destino en otro como se esquematiza en la Figura 1.2.

Por esta razón en el resto del documento siempre que tratemos con una red, y a no ser que se especifiquen otras propiedades, esta será dirigida, el grafo será como antes se ha especificado (simple y no un multigrafo) y siempre consideraremos implícitamente los nodos $s, t \in V$ con las propiedades antes mencionadas.

Por último, debemos hacer un apunte sobre los grafos conexos. Un grafo será *conexo* si entre cada par de nodos existe un conjunto de arcos, no necesariamente ordenados, que los unen y diremos que un grafo es *fuertemente conexo* si el conjunto de arcos está ordenado, es decir, existe un camino siguiendo las direcciones de los arcos entre ese par de nodos. En adelante supondremos grafos conexos y que existe al menos un camino desde el nodo s a todos los demás y desde todos los nodos al nodo t. Si se quisiera ampliar el concepto de conexión se recomienda ver Pelegrín et al. (1992, Cap. 2).

1.2. Flujo máximo y corte mínimo

Hemos definido que una red $R = (G, F)$ es un par constituido por un grafo $G = (V, A)$ y un conjunto de funciones F que están definidas en los arcos del grafo y cuyas imágenes son números reales. Fijaremos en el resto de la sección la función $c : A \to \mathbb{R}^+$ que nos indicará la capacidad máxima de cada arco de la red. Vamos a definir otra de estas funciones de la red R con una serie de restricciones adicionales que debe cumplir y que servirá para modelizar los problemas de flujo en redes. Esta será la función de flujo.

Definición 1.2.1. Un *flujo factible estático* sobre una red R es una función $x : A \to \mathbb{R}^+$ verificando:

- No negatividad: $x(a) \geq 0, \forall a \in A$.
- Restricción de capacidad: $x(a) \leq c(a), \forall a \in A$.
- Conservación de flujo:

$$\sum_{a \in \Gamma^{-1}(i)} x(a) = \sum_{a \in \Gamma(i)} x(a), \forall i \in V \backslash \{s, t\}.$$

En la definición incluimos la palabra *estático* ya que esta función no tiene restricciones temporales. Más adelante definiremos un flujo dinámico en el que introduciremos la componente temporal.

Diremos que el *valor de un flujo* es la cantidad de flujo que llega al nodo destino t. Por la condición de conservación de flujo se tiene que es la misma cantidad de flujo que sale del nodo fuente s:

$$v(x) := \sum_{a \in \Gamma^{-1}(t)} x(a) = \sum_{a \in \Gamma(s)} x(a). \tag{1.1}$$

Un problema directo que podemos resolver con las estructuras fijadas es el de la obtención de un flujo máximo en una red R dadas sus capacidades, esto es, maximizar el valor de $v(x)$. Como las condiciones de flujo factible son funciones lineales respecto del flujo x podemos plantear el problema de programación lineal continua siguiente con las variables continuas x_a el flujo en el arco $a \in A$:

$$\text{max.} \qquad v(x) \tag{1.2}$$

$$\text{s.a} \qquad x \text{ flujo factible en } R. \tag{1.2a}$$

En lugar de tener que resolver este problema mediante algún solver del mercado existen algoritmos que construyen iterativamente el flujo máximo. Los dos que vamos a ver aquí (Algoritmo 1 y Algoritmo 2) se deben a Ford y Fulkerson y se basan en partir de un flujo factible e ir actualizando su valor hasta conseguir un flujo máximo en la red.

Existe otra forma de encontrar un flujo máximo a partir del concepto de preflujo, esto es un flujo casi factible que no cumple la conservación del flujo en los nodos. En este caso, se parte de un preflujo máximo y en cada iteración se actualiza para intentar lograr la factibilidad. El lector puede encontrar el algoritmo demostrado en Goldberg y Tarjan (1988).

Primero introduciremos el concepto de corte de una red R, que es una partición disjunta del conjunto de nodos V del grafo original.

Definición 1.2.2. Sea $R = (V, A, c)$ una red y S, T una partición de V con $s \in S$ y $t \in T$. El conjunto de arcos incidentes en un nodo de S y en un nodo de T se denomina *corte* de R y se denota por $\langle S, T \rangle$. Un arco $(i, j) \in \langle S, T \rangle$ es un arco de ida si $i \in S$ y $j \in T$, es un arco de retorno si $j \in S$ y $i \in T$. Denotaremos al conjunto de arcos de ida por $\langle S, T \rangle^+$ y al conjunto de arcos de retorno por $\langle S, T \rangle^-$. La capacidad de un corte será la suma de las capacidades de sus arcos de ida y se denotará:

$$c\langle S, T \rangle = \sum_{a \in \langle S,T \rangle^+} c(a).$$

Observación. Si usamos los conjuntos de arcos sucesores y predecesores para definir un corte de manera conjuntista tendremos que:

$$\langle S,T\rangle^+ := \{\Gamma(i) : i \in S\} \cap \{\Gamma^{-1}(i) : i \in T\},$$

$$\langle S,T\rangle^- := \{\Gamma^{-1}(i) : i \in S\} \cap \{\Gamma(i) : i \in T\},$$

$$\langle S,T\rangle := \langle S,T\rangle^+ \cup \langle S,T\rangle^-.$$

El problema del corte mínimo en una red será el de encontrar un corte $\langle S,T\rangle$ con capacidad mínima. Veremos la relación que tiene con un flujo factible definiendo el flujo que pasa por un corte:

Definición 1.2.3. Dado un flujo x y un corte $\langle S,T\rangle$ de una red R, el *flujo a lo largo del corte* será la suma de los flujos de los arcos de ida menos la suma del flujo de los arcos de retorno y se denotará por:

$$x\langle S,T\rangle := \sum_{a\in\langle S,T\rangle^+} x(a) - \sum_{a\in\langle S,T\rangle^-} x(a).$$

Proposición 1.2.4. *Sea R una red dirigida. Para cualquier flujo factible x y cualquier corte $\langle S,T\rangle$ sobre R se tiene que $v(x) = x\langle S,T\rangle$.*

Demostración. Sea $\langle S,T\rangle$ un corte sobre la red R, como x es flujo factible, por la Definición 1.2.1 se tiene que:

$$v(x) = \sum_{a\in\Gamma(s)} x(a) \qquad \text{y} \qquad \sum_{a\in\Gamma^{-1}(i)} x(a) = \sum_{a\in\Gamma(i)} x(a) \quad \forall i \in S\backslash\{s\}.$$

Por tanto,

$$v(x) = \sum_{a\in\Gamma(s)} x(a) + \sum_{i\in S\backslash\{s\}} \left(\sum_{a\in\Gamma^{-1}(i)} x(a) - \sum_{a\in\Gamma(i)} x(a) \right),$$

en donde el segundo sumatorio es nulo por conservación del flujo x. Fijémonos que en los sumatorios anteriores los términos cuyos arcos entran y salen de un nodo del conjunto S se anulan dos a dos y solo nos quedan los términos cuyos arcos entran a T o bien salen de T, es decir:

$$v(x) = \sum_{a\in\langle S,T\rangle^+} x(a) - \sum_{a\in\langle S,T\rangle^-} x(a).$$

□

La anterior proposición es clave para la relación entre corte y flujo en una red, pues nos dice que el valor de un flujo factible es igual al flujo a lo largo de cualquier corte sobre esa misma red. Gracias a esto podemos probar los siguientes corolarios.

Corolario 1.2.5. *Sea x un flujo factible sobre una red R y $\langle S,T\rangle$ un corte sobre dicha red, entonces, $v(x) \leq c\langle S,T\rangle$.*

Demostración. Por la Definición 1.2.1 un flujo factible x cumple que $0 \leq x(a) \leq c(a) \ \ \forall a \in A$, y por la proposición anterior se tiene:

$$v(x) = x\langle S,T\rangle = \sum_{a\in\langle S,T\rangle^+} x(a) - \sum_{a\in\langle S,T\rangle^-} x(a) \leq \sum_{a\in\langle S,T\rangle^+} c(a) = c\langle S,T\rangle.$$

□

Corolario 1.2.6. *Sea x un flujo factible sobre una red R y $\langle S,T\rangle$ un corte sobre dicha red. Supongamos que $v(x) = c\langle S,T\rangle$. Entonces x es un flujo máximo sobre R y $\langle S,T\rangle$ es un corte mínimo.*

Demostración. Supongamos que $v(x) = c\langle S,T\rangle$. Por el corolario anterior sabemos que cualquier flujo factible está acotado superiormente por la capacidad de un corte cualquiera, por tanto, el valor del flujo x alcanza su máximo. Análogamente, por el corolario anterior sabemos que cualquier corte de una red está acotado inferiormente por el valor de un flujo factible cualquiera, por el tanto, la capacidad del corte $\langle S,T\rangle$ alcanza su mínimo.

En otras palabras, estamos diciendo que el corte $\langle S,T\rangle$ es un cuello de botella para el flujo x en la red R. □

A continuación, introduciremos una notación para decir explícitamente el sentido de los arcos en la red. Sea $a = (i,j) \in A$ con $i,j \in V$, entonces denotaremos por a^+ al arco en el sentido original (i,j) y por a^- al arco en sentido contrario (j,i). Si bien es cierto que no siempre existen arcos en los dos sentidos en una red dada vamos a construir una red a partir de un flujo factible en la que necesitaremos distinguir entre arcos de ida y de retorno.

Definición 1.2.7. Dado un flujo factible x sobre una red dirigida $R = (V,A,c)$ la *red residual* de x viene dada por $R_x = (V, A_x, c_x)$ donde

- $a^+ \in A_x$ si $x(a) < c(a)$ con $c_x(a^+) = c(a) - x(a)$.
- $a^- \in A_x$ si $x(a) > 0$ con $c_x(a^-) = x(a)$.

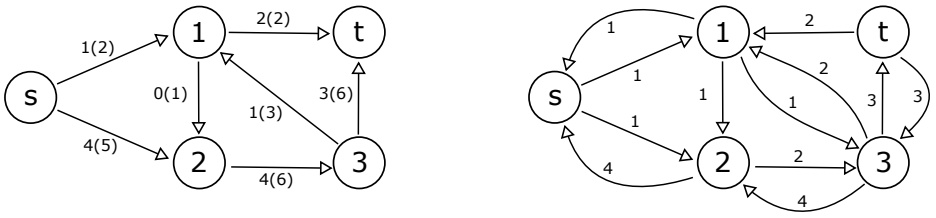

Figura 1.3: Red R con flujo factible x y las capacidades entre paréntesis (izquierda) y red residual R_x con las capacidades c_x (derecha)

Hemos supuesto que en nuestra red siempre existirá al menos un camino P entre el nodo s y el nodo t. La pregunta sería, cuándo sigue existiendo un camino entre s y t en la red residual dado un flujo factible x. El Algoritmo 1 se basa en el hecho de que si no se encuentra P, entonces el flujo x será máximo.

Definición 1.2.8. Dado un flujo factible x sobre una red dirigida $R = (V, A, c)$, un *camino de aumento de flujo* es un camino P de s a t en la red residual R_x, es decir, un subconjunto ordenado de A_x. La capacidad de dicho camino será $\Delta_P = \min_{a \in P}\{c_x(a)\}$.

Observación. La capacidad será la mínima de todas las capacidades ya que por ese camino querremos mandar un flujo extra y, por tanto, tiene que respetar todas las capacidades, en particular, el mayor valor de flujo que se puede mandar a través del camino P es la mínima de todas las capacidades de los arcos que lo componen. El número de caminos de aumento de flujo que podemos llegar a encontrar está acotado por $n \cdot \max_a\{c(a)\}$, siendo $n = |V|$.

Usando la red residual de la Figura 1.3 podemos construir el siguiente camino de aumento de flujo $P = \{(s,1),(1,3),(3,t)\}$ con $\Delta_P = 1$.

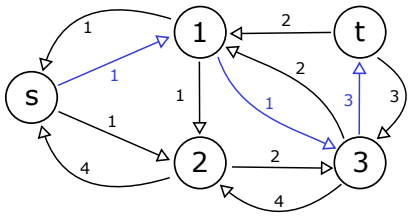

Figura 1.4: Camino de aumento P (en azul) en la red residual R_x

Ahora usaremos el camino de aumento de flujo P para incrementar el valor del flujo x original. Posteriormente probaremos que de no existir dicho camino P entonces el valor de x no se puede aumentar.

Proposición 1.2.9. *Dado un flujo factible x sobre una red dirigida $R = (V, A, c)$ y P un camino de aumento de flujo con capacidad Δ_P, definiendo el siguiente flujo:*

$$x^P(a) := \begin{cases} x(a) + \Delta_P & \text{si } a^+ \in P \text{ y } x(a) < c(a) \\ x(a) - \Delta_P & \text{si } a^- \in P \text{ y } x(a) > 0 \\ x(a) & \text{en otro caso,} \end{cases}$$

se tiene que x^P es un flujo factible sobre R y $v(x^P) = v(x) + \Delta_P$.

Demostración. Empecemos viendo que el valor del flujo es como dice el enunciado:

$$v(x^P) := \sum_{a \in \Gamma(s)} x^P(a) := \sum_{a \in \Gamma(s)} x(a) + |\{a \in \Gamma(s) : a^+ \in P\}|\Delta_P - |\{a \in \Gamma(s) : a^- \in P\}|\Delta_P.$$

Estas igualdades se siguen de la definición de valor de flujo y de observar que el camino P que va de s hasta t en la red residual R_x puede salir del nodo s una cantidad de veces $|\{a \in \Gamma(s) : a^+ \in P\}|$ y entrar al nodo s una cantidad de veces $|\{a \in \Gamma(s) : a^- \in P\}|$ pero al tener que ser un camino hasta t, el primer conjunto tiene necesariamente un arco más que el segundo conjunto y nos queda:

$$v(x^P) = \sum_{a \in \Gamma(s)} x(a) + \Delta_P = v(x) + \Delta_P.$$

Ahora veremos que el flujo x^P es factible usando la Definición 1.2.1. Primero procedemos sobre las condiciones de capacidad del flujo:

- Sea $a \in A$ tal que $a^+ \in P \subset A_x$. Entonces $x(a) < c(a)$ pues existe $a^+ \in A_x$ y $c_x(a^+) = c(a) - x(a)$. Supongamos que $x^P(a) > c(a)$, entonces por la definición de x^P se tendría que $\Delta_P > c(a) - x(a) = c_x(a^+)$, lo cual es una contradicción pues $\Delta_P = \min_{a \in P}\{c_x(a)\}$. Además, como $\Delta_P \geq 0$ es evidente que $x^P(a) \geq 0$.
- Sea $a \in A$ tal que $a^- \in P \subset A_x$. Entonces $x(a) > 0$ pues existe $a^- \in A_x$ y $c_x(a^-) = x(a)$. Supongamos que $x^P(a) > c(a)$ entonces $x(a) - c(a) > \Delta_P$, como x es flujo factible se tiene que $x(a) \leq c(a)$ y por tanto $0 > \Delta_P$ lo cual es una contradicción. Por otro lado, supongamos que $x^P(a) < 0$ entonces se tiene que $\Delta_P > x(a) = c_x(a^-)$, lo cual es una contradicción pues $\Delta_P = \min_{a \in P}\{c_x(a)\}$.
- Sea $a \in A$ tal que $a^-, a^+ \notin P$, es decir, un arco por el que no pasa el camino de aumento de flujo, como aquí el flujo no cambia se tiene que $0 \leq x^P(a) = x(a) \leq c(a)$.

Falta ver la conservación de flujo en todos los nodos de la red. Fijemos $i \in V \backslash \{s, t\}$. Entonces:

$$\sum_{a \in \Gamma^{-1}(i)} x^P(a) = \sum_{a \in \Gamma^{-1}(i)} x(a) + |\{a \in \Gamma^{-1}(i) : a^+ \in P\}|\Delta_P - |\{a \in \Gamma^{-1}(i) : a^- \in P\}|\Delta_P,$$

$$\sum_{a \in \Gamma(i)} x^P(a) = \sum_{a \in \Gamma(i)} x(a) + |\{a \in \Gamma(i) : a^+ \in P\}|\Delta_P - |\{a \in \Gamma(i) : a^- \in P\}|\Delta_P.$$

Fijémonos en que el número de arcos del camino P que entran al nodo i es exactamente la suma de los cardinales $|\{a \in \Gamma^{-1}(i) : a^+ \in P\}| + |\{a \in \Gamma(i) : a^- \in P\}|$ y el número de arcos que salen del nodo i que son del camino P es la suma de los cardinales $|\{a \in \Gamma^{-1}(i) : a^- \in P\}| + |\{a \in \Gamma(i) : a^+ \in P\}|$. Por tanto, por la conservación de flujo de x, los dos sumatorios anteriores son iguales y se tiene que x^P conserva el flujo en la red R. □

Si ahora usamos el camino de aumento de flujo de la Figura 1.4 con $\Delta_P = 1$ sobre el flujo factible de la Figura 1.3 con valor $v(x) = 5$ podemos obtener un flujo x^P con un valor $v(x^P) = 6$ tal y como dice la proposición anterior. El flujo actualizado sería:

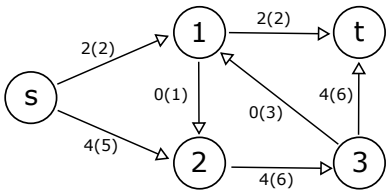

Figura 1.5: Red R con flujo factible aumentado x^P

Teorema 1.2.10. *Sea x un flujo factible sobre una red dirigida $R = (V, A, c)$. Entonces x es flujo máximo si y solo si no existe ningún camino de aumento de flujo en la red residual R_x.*

Demostración. $\boxed{\Longrightarrow}$ Es trivial por el contrarrecíproco, pues si existe un camino de aumento de flujo P entonces podemos definir el flujo x^P como en la proposición anterior y su valor es mayor que el valor del flujo x, por tanto, x no podía ser un flujo máximo.

$\boxed{\Longleftarrow}$ Vamos a definir un conjunto de nodos como sigue. Sea $S = \{s\}$ y añadiremos el nodo $j \in V$ al conjunto si $\exists i \in S$ tal que $x(i,j) < c(i,j)$ o $x(i,j) > 0$.

Evidentemente, $t \notin S$ pues si perteneciese habríamos encontrado un camino de aumento de flujo. Definimos el conjunto de nodos no vacío $T = V \backslash S$. Consideremos el corte de la red R dado por $\langle S, T\rangle$. Entonces por construcción de S tenemos:

- Si $a \in \langle S,T\rangle^+$ entonces $x(a) > 0$ pero no puede ser $x(a) < c(a)$, por tanto, $x(a) = c(a)$.
- Si $a \in \langle S,T\rangle^-$ entonces $x(a) < c(a)$ pero no puede ser $x(a) > 0$, por tanto, $x(a) = 0$.

Calculando el valor del flujo a través del corte que hemos definido se tiene:

$$v(x) = x\langle S,T\rangle = \sum_{a\in\langle S,T\rangle^+} x(a) - \sum_{a\in\langle S,T\rangle^-} x(a) = \sum_{a\in\langle S,T\rangle^+} c(a) = c\langle S,T\rangle.$$

Por el Corolario 1.2.6 el flujo x es máximo. □

Con la teoría dada, estamos en disposición de describir el algoritmo de Ford-Fulkerson que data de 1956 (ver Ford Jr y Fulkerson, 1956). Este se basa en tomar un flujo inicial x_0 factible sobre una red $R = (V, A, c)$ que se irá actualizando usando la red residual R_{x_0} y algún camino de aumento de flujo en la misma. Una explicación con una notación similar a la utilizada aquí la encontramos en Ahuja et al. (1993, p. 180-184) en la que podemos ver además algunos ejemplos y reflexiones que da el autor.

Algoritmo 1 Ford-Fulkerson $\mathcal{O}(n^3)$

Input: Red dirigida R, flujo factible x_0
Pasos:

1. Hacemos $k = 0$.
2. Construimos la red residual R_{x_k}.
3. Buscamos un camino P de s a t en R_{x_k}.
4. Si no existe P entonces x_k es flujo máximo, FIN.
5. Si existe P, actualizamos el flujo

$$x_k(a) = \begin{cases} x_k(a) + \Delta_P & \text{si } a^+ \in P \text{ y } x_k(a) < c(a) \\ x_k(a) - \Delta_P & \text{si } a^- \in P \text{ y } x_k(a) > 0 \\ x_k(a) & \text{en otro caso.} \end{cases}$$

6. Hacemos $k = k + 1$ y volvemos al paso 2.

El flujo factible inicial puede ser el trivial idénticamente nulo o cualquiera que respete las limitaciones de la función c de la red. Alternativamente existe un algoritmo que sustituye el proceso de construir la red residual y buscar un camino de aumento de flujo directamente en cada iteración por un proceso de etiquetado (ver Ford Jr y Fulkerson, 1962, p. 17-18).

Consideremos etiquetas (δ_i, Δ_i) en cada nodo $i \in V$, donde δ_i denota el nodo del que procede la etiqueta y el sentido de la misma (por avance con $+$ o por retroceso con $-$) y Δ_i es la cantidad máxima de aumento de flujo que se puede hacer hasta el nodo i. Este algoritmo al terminar encuentra un camino de aumento de flujo de manera implícita simplemente recorriendo las etiquetas desde t hasta s según se indique avance o retroceso y cuyo valor de aumento será Δ_t.

Algoritmo 2 Ford-Fulkerson (etiquetado) $\mathcal{O}(n^3)$

Input: Red dirigida R, flujo factible x_0
Pasos:

1. Hacemos $k = 0$.
2. Etiquetamos el nodo s como $(-, \infty)$.
3. Tomamos un arco $a \in A$ que conecta dos nodos $i, j \in V$:
   - ▷ Si $a = (i, j)$ y $x_k(a) < c(a)$ entonces etiquetamos j por avance con (i^+, Δ_j),
   $$\Delta_j = \min\{\Delta_i, c(a) - x_k(a)\}.$$
   - ▷ Si $a = (j, i)$ y $x_k(a) > 0$ entonces etiquetamos j por retroceso con (i^-, Δ_j),
   $$\Delta_j = \min\{\Delta_i, x_k(a)\}.$$
4. Repetimos el paso 3 hasta llegar a una de las dos situaciones siguientes:
   - ▷ El nodo t ha sido etiquetado, continuamos al paso 5
   - ▷ El nodo t no ha podido ser etiquetado, x_k es flujo máximo, FIN.
5. Tenemos que δ_t representa al predecesor de t. Construimos iterativamente el camino de aumento de flujo P de s a t.
6. Para cada arco $a = (i, j) \in P$:
   - ▷ Si $\delta_j = i^+$ entonces $x_k(a) = x_k(a) + \Delta_t$
   - ▷ Si $\delta_j = i^-$ entonces $x_k(a) = x_k(a) - \Delta_t$
7. Hacemos $k = k + 1$ y volvemos al paso 2.

Este proceso de etiquetado permite automatizar el algoritmo en un ordenador sin tener que utilizar subrutinas complementarias que calculen caminos y construyan la red residual. Sin embargo, el Algoritmo 1 en ocasiones es más visual que el proceso de etiquetado desarrollado en el Algoritmo 2. En ambos casos tendremos una complejidad computacional de $\mathcal{O}(n^3)$ siendo n el número de nodos de la red.

Como colofón de esta sección demostraremos la relación íntima que guardan el problema del flujo máximo en una red y el problema del corte mínimo (ver Definición 1.2.2).

Teorema 1.2.11 (Flujo Máximo - Corte Mínimo)**.** *El valor de un flujo máximo de una red dirigida es igual a la capacidad de un corte mínimo de la red.*

Demostración. Supongamos que x es un flujo máximo en la red R. Por el Teorema 1.2.10 sabemos que no existe un camino de aumento de flujo en la red residual R_x y, como vimos en su demostración, se puede construir un corte $\langle S, T\rangle$ de R tal que el valor del flujo es igual a la capacidad del corte, por el Corolario 1.2.6 dicho corte es mínimo.

Supongamos que tenemos un corte $\langle S, T\rangle$ de R con capacidad mínima, entonces construimos un flujo factible x, en particular que cumpla la conservación de flujo, tal que:

$$x(a) = \begin{cases} c(a) & \text{si } a \in \langle S, T\rangle^+ \\ 0 & \text{si } a \in \langle S, T\rangle^- \\ x(a) & \text{factible en otro caso.} \end{cases}$$

Podemos construirlo a partir de un flujo factible cualquiera y usando caminos de aumento de flujo iterativamente. Al ser el corte $\langle S, T\rangle$ de capacidad mínima, será el primer corte de la red R que se sature completamente, es decir, en el que se obtenga un flujo de la manera que hemos indicado. Dicho flujo será máximo pues su valor en el corte $\langle S, T\rangle$ coincide con la capacidad del corte (ver Corolario 1.2.6). □

1.3. Flujo a coste mínimo

Hemos definido un flujo factible en una red dada R y hemos resuelto el problema de encontrar aquellos que tengan un valor máximo. Aunque el flujo máximo nos sirve para modelizar muchos problemas de la realidad, estos no capturan una variable esencial como es el *coste*. No hay problema de optimización en la realidad sin un coste asociado, ya sea temporal o monetario.

En esta sección incluiremos en nuestra red $R = ((V, A), c)$, con capacidades en los arcos, una función $\tau : A \to \mathbb{R}$, que designa el coste por unidad de flujo sobre un arco. Dejaremos que el coste pueda ser menor que cero ya que, para desarrollar la matemática que necesitamos, algunos arcos de nuestra red podrán tener un coste negativo. La interpretación no es otra que favorecer el paso del flujo por esos arcos ya que no suponen un coste, es más, suponen un beneficio si son estrictamente negativos.

El problema que nace de esta red $R = ((V, A), (c, \tau))$ es el de encontrar aquel flujo factible, con un valor $v \in \mathbb{R}$ dado a priori, que tenga coste mínimo. El coste del flujo vendrá dado por la suma del flujo que pasa por los arcos multiplicado por el coste $\tau(a)$. En particular, podremos resolver el problema de encontrar el flujo factible estático de valor máximo a coste mínimo si el valor v ha sido calculado mediante un algoritmo de flujo máximo.

Si planteamos la cuestión del párrafo anterior como un problema de programación lineal quedaría:

$$\text{min.} \quad \sum_{a\in A} \tau_a x_a \tag{1.3}$$

$$\text{s.a} \quad x \text{ flujo factible en } R \tag{1.3a}$$

$$v(x) = v \text{ constante.} \tag{1.3b}$$

Como en el caso anterior, existen numerosos algoritmos que resuelven este problema. Nosotros explicaremos el que se debe a Busacker y Gowen (ver Busacker y Gowen, 1961). Este resuelve el problema anterior de mínimo coste con un flujo que va desde el nodo s hasta el nodo t que hemos fijado. Sin embargo, existe otra versión del problema en la que no se destacan dos nodos especiales como nosotros hacemos y lo que se tiene es una red con un flujo circulando a coste mínimo sin origen ni destino. Uno de los algoritmos más populares para esta versión es el algoritmo "Out-of-kilter" debido a Fulkerson en 1961 y podemos encontrar una explicación del mismo en Pelegrín et al. (1992, p. 177-183).

Algoritmo 3 Busacker-Gowen $\mathcal{O}(n^2m)$

Input: Red dirigida $R = (V, A, c, \tau)$, valor $v \in \mathbb{R}$

Pasos:

1. Construimos el flujo x_0 nulo. Hacemos $k = 0$.
2. Construimos la red residual $R_{x_k} = (V, A_{x_k}, c_{x_k}, \tau_{x_k})$ donde, análogamente a la Definición 1.2.7, tendremos:
   - ▷ $a^+ \in A_{x_k}$ si $x_k(a) < c(a)$ con $c_{x_k}(a^+) = c(a) - x_k(a)$, $\tau_{x_k}(a^+) = \tau(a)$.
   - ▷ $a^- \in A_{x_k}$ si $x_k(a) > 0$ con $c_{x_k}(a^-) = x_k(a)$, $\tau_{x_k}(a^-) = -\tau(a)$.
3. Obtenemos el camino de coste mínimo P_k de s a t en la red residual R_{x_k}. Si el camino no existe, no hay solución, FIN.
4. Calculamos $\epsilon = \min\{v, \{c_{x_k}(a) : a \in P_k\}\}$ y actualizamos el flujo x_k como:

$$x_k(a) = \begin{cases} x_k(a) + \epsilon & \text{si } a^+ \in P_k \text{ y } x_k(a) < c(a) \\ x_k(a) - \epsilon & \text{si } a^- \in P_k \text{ y } x_k(a) > 0 \\ x_k(a) & \text{en otro caso.} \end{cases}$$

5. Actualizamos $v = v - \epsilon$. Si $v > 0$ hacemos $k = k + 1$ y volvemos al paso 2. En caso contrario, FIN, x_k es flujo a coste mínimo.

Observación. Atendamos a que, si en el algoritmo eliminamos las condiciones referentes al valor de flujo v y simplemente vamos calculando caminos de aumento de flujo a coste mínimo en cada iteración, al final obtendremos un **flujo máximo a coste mínimo** pues estamos forzando a saturar todos los caminos de aumento que se encuentren.

Este algoritmo se basa, como el en el caso del Algoritmo 1, en tomar un camino en la red residual y actualizar el flujo en cada iteración según las capacidades de los arcos del camino encontrado y la constante v. El algoritmo es de hace casi 60 años y ha sufrido algunas mejoras en todo este tiempo. Nosotros destacaremos la realizada en Goldberg y Tarjan (1987). Aquí consiguen un algoritmo cuya complejidad computacional es menor en el peor de los casos que el Algoritmo 3. Si n es el número de nodos y m el de arcos entonces es conocido que la complejidad computacional en el peor de los casos es $\mathcal{O}(n^2m)$. Para una explicación del concepto de complejidad computacional se recomienda al lector mirar el Anexo A.1 y las lecturas citadas allí.

Vemos que el Algoritmo 3 encuentra un camino P_k a coste mínimo en la red residual R_{x_k}. Para hacerlo tendrá que usar alguno de los algoritmos de caminos más cortos entre dos nodos que existen, ya que los costes pueden ser interpretados como distancias y, por tanto, el camino a coste mínimo se interpretaría como un camino más corto.

Uno de los algoritmos clásicos más conocidos para caminos más cortos es el Algoritmo de Dijkstra (ver Dijkstra, 1959), que calcula el camino más corto entre dos nodos si todas las distancias son mayores o iguales que cero. Como en nuestro caso en el Algoritmo 3 definimos un coste que toma valores negativos para algunos arcos, este no nos sirve en esta ocasión.

Otro de los algoritmos más conocidos es el de Floyd (ver Floyd, 1962) que sí que permite distancias negativas pero que calcula los caminos más cortos entre todos los pares de nodos actualizando en cada iteración una matriz de distancias. Esto supone un sobrecoste computacional ya que nosotros solo necesitamos el camino más corto entre dos nodos. Por ello consideraremos el Algoritmo de Moore (ver Moore, 1959). Este es muy similar al de Dijkstra pero permite distancias negativas durante su ejecución. En Pelegrín et al. (1992, p. 90-91) podemos encontrar el Algoritmo de Moore. El capítulo 4 de ese mismo libro está dedicado expresamente a caminos más cortos en un grafo y podemos ver, aparte de los citados aquí, otros algoritmos para estructuras de grafos concretas.

1.4. Flujo y localización

Cuando hablábamos de las aplicaciones de las redes en la Sección 1.1 sobre problemas de localización y problemas de flujo, lo hicimos por separado. En las Secciones 1.2 y 1.3 hemos estado describiendo los problemas del flujo máximo en una red y el flujo a coste mínimo, ambos estáticos sin considerar el tiempo. En numerosas ocasiones los problemas de localización plantean encontrar puntos de la red que minimicen alguna función de la distancia entre todos los nodos, que estarían modelando zonas de demanda. Un par de ejemplos de problemas de localización muy estudiados son el problema de la p-mediana y el p-centro, los cuales se diferencian en la función objetivo que intentan minimizar. Si el lector quiere conocer más sobre este tipo de problemas y realizar un recorrido histórico le recomendamos ver Marín y Pelegrín (2019) y Çalık et al. (2019).

Sin embargo, es posible que, dada la naturaleza de algunos problemas, tengamos que localizar alguna instalación/objeto en una red (ya sea un nodo o un arco de la misma) y a la vez tengamos que buscar la localización para dicha instalación que afecte lo menos posible a un

flujo que circule por la red, digamos de s a t. Nace de esta manera la necesidad de los modelos de flujo y localización simultánea que denominaremos, como en la terminología anglosajona, *Problemas q-FlowLoc* a partir de ahora, siendo q el **número de instalaciones** u objetos que se quieren colocar en toda la red.

Así, por ejemplo, los problemas de evacuación modelados con una red pueden ser vistos en dos fases. La primera sería la localización de puestos de emergencia a lo largo de la red, los cuales deben dificultar lo menos posible el paso de la gente, y la segunda fase sería la maximización del flujo de personas a evacuar, es decir, tenemos un problema *FlowLoc* según hemos descrito. Incluso, si nos referimos a la planificación de una evacuación de un instituto, un estadio o una zona urbana, es importante no situar instalaciones (ya sean bancos, expositores, puestos de comida, etc) en lugares que hagan disminuir el flujo máximo de una manera dramática, ya que sería un riesgo para una posible evacuación del lugar en cuestión.

En esta sección consideraremos inicialmente una red $R = (V, A, c)$ con dos nodos destacados s y t de origen y destino del flujo que haremos pasar por la red. Las instalaciones solo se podrán colocar en arcos de la red. Es posible que el problema que se quiera modelizar en la red tenga necesariamente más de un nodo fuente y más de un nodo destino. En Heller y Hamacher (2011) se trata esta extensión del problema al que llaman "multiterminal q-FlowLoc". Nosotros consideraremos la simplificación del modelo en la cual acumularemos todos los nodos fuente y destino en un solo par s y t como vimos en la Figura 1.2.

Definición 1.4.1. Sean una red $R = (V, A, c)$, s el nodo fuente y t el nodo destino. Consideremos $\mathbb{P} = \{p_1, p_2, \ldots, p_q\}$ el conjunto de instalaciones a localizar en los arcos de la red, y las siguientes funciones:

$$d : \mathbb{P} \to \mathbb{R}^+,$$
$$m : A \to \mathbb{R}^+,$$

siendo d una función que asigna el tamaño o dimensiones de nuestra instalación. La función m indica el número máximo de instalaciones que se pueden localizar en un arco $a \in A$, y permite definir el conjunto $\mathbb{L} := \{a \in A : m(a) > 0\}$ que será el conjunto de localizaciones factibles de nuestra red. Denotaremos por ***red de flujo y localización*** o ***red de evacuación*** a la red

$$R_{\mathbb{P}} = ((V, A), (c, d, m)).$$

A partir de la definición de red de flujo y localización nace, de manera natural, el concepto de función de localización. Éste asignará a cada instalación del conjunto $\mathbb{P}$ un arco $a \in A$ que pueda albergar, al menos, una localización. Esto es que $m(a) > 0$, es decir, $a \in \mathbb{L}$.

Definición 1.4.2. Sea $R_{\mathbb{P}}$ una red de flujo y localización, llamaremos **función de localización** de $\mathbb{P}$ a la función

$$l : \mathbb{P} \to \mathbb{L},$$

que asigna a cada instalación $p \in \mathbb{P}$ una única localización de $\mathbb{L}$.

Observación. Notemos que, aunque una instalación no se puede localizar en dos sitios diferentes, dos instalaciones distintas sí pueden compartir la misma localización. Esto dependerá del valor de la función $m(a)$ para un arco a (que representa la localización).

Entonces nos podemos plantear el problema de flujo y localización que hemos descrito en los párrafos anteriores y cuyas características reuniremos de la siguiente manera:

Problema q-FlowLoc

Input: Una red de flujo y localización $R_{\mathbb{P}} = ((V, A), (c, d, m))$.

Se pide: Encontrar una función de localización $l : \mathbb{P} \to \mathbb{L}$, tal que el flujo de s a t sea máximo en la red modificada

$$R_l := (V, A, c_l) \quad \text{donde} \quad c_l(a) := c(a) - \max_{p \in \mathbb{P}} \{d(p) \ : \ l(p) = a\}$$

son las capacidades de los arcos modificadas por la asignación que proporciona la función de localización.

Observación. El hecho de usar la función de coste nueva c_l, calculada como el máximo de todas las funciones de tamaño de las instalaciones en un arco dado, es porque así se está modelizando que solo la instalación de mayor tamaño es la que causa una disminución de flujo. Por ejemplo, en la Figura 1.6 se puede ver que la capacidad de un arco a se verá reducida por la instalación p_i de la imagen que tenga el mayor de los tamaños, provocando un "cuello de botella".

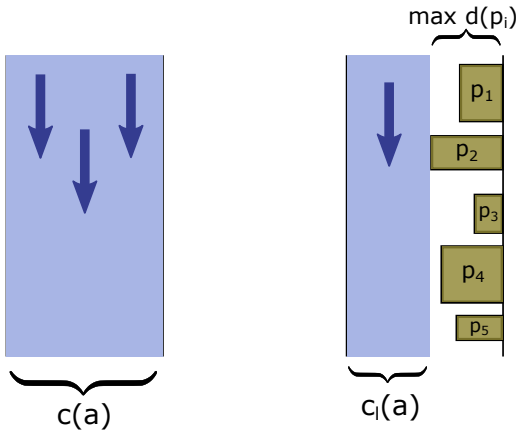

Figura 1.6: A la izquierda, el flujo (azul) por un arco a. A la derecha, el mismo arco con unas instalaciones colocadas

Observación. Sin embargo, es posible que en otros contextos queramos escribir $c_l(a) = c(a) - \sum_{p:l(p)=a} d(p)$. De esta manera las instalaciones "molestan" más al flujo cuantas más se acumulen en un mismo arco. Por ejemplo, el caso del atraque de una serie de barcos en un canal cuando estos se tienen que colocar en el mismo punto por la falta de espacio y el tamaño de la instalación es la suma de todos los tamaños individuales (Ver Figura 1.7, izquierda). También es posible entender este modelo en el caso en el que por un tubo queramos hacer pasar una serie de cables de distintos tamaños cada uno, algo muy común en instalaciones eléctricas domésticas. Entonces el flujo es todo el que pueda pasar por el espacio libre que nos queda en el tubo, mientras que cada cable (que será una instalación p) irá acumulando en la disminución de la capacidad (ver Figura 1.7, derecha).

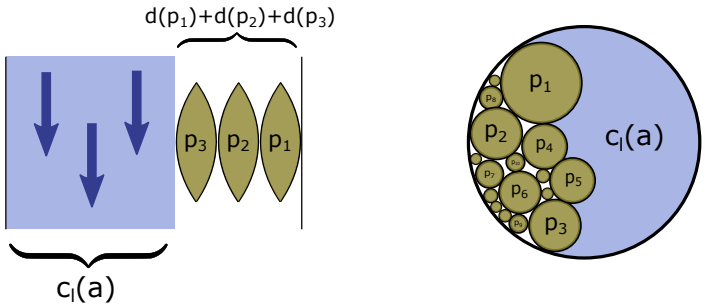

Figura 1.7: A la izquierda, el flujo (azul) en un canal con 3 barcos abarloados. A la derecha, un flujo (azul) a través de un tubo con varios cables ya colocados

Plantearemos el problema mediante su formulación lineal entera mixta. Después motivaremos la creación de una heurística para el caso más general debido a que este problema es $\mathcal{NP}$-completo (ver Teorema 1.4.3).

Variables

x_a : flujo en el arco a $\quad \forall a \in A$

$y_{a,p}$: $\begin{cases} 1 \text{ si la instalación } p \text{ se coloca en el arco } a \\ 0 \text{ en otro caso.} \end{cases}$ $\quad \forall a \in A, p \in \mathbb{P}$

Constantes

d_p : tamaño de la instalación $p \in \mathbb{P}$.

c_a : capacidad del arco $a \in A$.

m_a : máximo número de instalaciones que se pueden colocar en $a \in A$.

Formulación

$$\text{max.} \quad \sum_{a \in \Gamma(s)} x_a - \sum_{a \in \Gamma^{-1}(s)} x_a \tag{1.4}$$

$$\text{s.a} \quad \sum_{a \in \Gamma^{-1}(i)} x_a - \sum_{a \in \Gamma(i)} x_a = 0 \qquad \forall i \in V \backslash \{s,t\} \tag{1.4a}$$

$$0 \leq x_a \leq c_a \qquad \forall a \in A \backslash \mathbb{L} \tag{1.4b}$$

$$\sum_{p \in \mathbb{P}} y_{a,p} \leq m_a \qquad \forall a \in \mathbb{L} \tag{1.4c}$$

$$\sum_{a \in \mathbb{L}} y_{a,p} = 1 \qquad \forall p \in \mathbb{P} \tag{1.4d}$$

$$x_a + d_p y_{a,p} \leq c_a \qquad \forall a \in \mathbb{L}, p \in \mathbb{P} \tag{1.4e}$$

$$y_{a,p} \in \{0,1\} \qquad \forall a \in \mathbb{L}, p \in \mathbb{P} \tag{1.4f}$$

En el objetivo (1.4) tratamos de maximizar el valor del flujo; el segundo sumatorio se utiliza para evitar usar arcos que lleguen hasta el nodo s. Si no existen, es decir, si $\Gamma^{-1}(s) = \emptyset$ entonces no se computaría el segundo sumatorio y nos quedaría maximizar $v(x)$ en un flujo estático. Otra forma de conseguir que este sumatorio sea siempre nulo es, si es posible en el modelo, crear un nodo fuente y un nodo destino que acapare al resto en uno solo como ya vimos en la Figura 1.2.

Los conjuntos de restricciones (1.4a) y (1.4b) permiten hacer que x sea un flujo factible estático en la red (véase Definición 1.2.1). Las restricciones (1.4c) aseguran que cada localización válida en un arco, cuyo conjunto es $\mathbb{L} \subset A$, no supere el máximo de instalaciones posibles en dicho arco dado por valor m_a. Las restricciones (1.4d) obligan a que cada instalación p sea colocada en una y solo una localización válida. Las restricciones (1.4e) se encargan de que si una instalación p se coloca en un arco a, el flujo que pasa por ese arco no pueda superar a la capacidad menos el tamaño de la instalación d_p. Al ser las variables $y_{a,p}$ binarias, se logra que el flujo del arco a no supere a la capacidad menos el máximo de los tamaños de todas las instalaciones situadas en a. Por último el conjunto de restricciones (1.4f) nos indican que las variables $y_{a,p}$ son binarias, las variables x_a son continuas y por ello no se explicitan en la formulación.

Con la notación que hemos fijado hasta ahora $|\mathbb{P}| = q$. Empezaremos elaborando un algoritmo exacto para el caso en el que queramos localizar únicamente una instalación $\mathbb{P} = \{p\}$ con $q = 1$. Una primera aproximación a resolver el Problema q-FlowLoc es por enumeración, ya que queremos localizar una sola instalación y lo único que tendremos que hacer es buscar la localización que afecte lo menos posible al flujo máximo original. Para ello iremos recolocando nuestra instalación p y calculando el flujo máximo para cada arco $a \in \mathbb{L}$ entre las localizaciones factibles. Nos quedaremos entonces con el flujo máximo entre todos los calculados para el conjunto $\mathbb{L}$ y la localización para la que se alcance. El siguiente algoritmo es un resumen de lo expuesto en este párrafo.

Algoritmo 4 1-FlowLoc por enumeración $\mathcal{O}(|\mathbb{L}|n^3)$

Input: Red de flujo y localización $R_{\mathbb{P}} = ((V, A), (c, d, m))$ e instalación p
Pasos:

1. Fijamos el flujo máximo $v = -\infty$.
2. Tomamos un arco $a \in \mathbb{L}$ no marcado.
3. Si $c(a) \geq d(p)$ entonces actualizamos $c(a) = c(a) - d(p)$.
4. Calculamos el flujo máximo x^a en la red $R = (V, A, c)$ con el Algoritmo 2.
5. Marcamos el arco a y reactualizamos la capacidad $c(a) = c(a) + d(p)$.
6. Si $v(x^a) > v$ entonces actualizamos el flujo máximo $v = v(x^a)$ y la localización $l(p) = a$.
7. Si todos los arcos de $\mathbb{L}$ están marcados, FIN. Si no, volvemos al paso 2.

En Hamacher et al. (2013, p. 167-171) podemos encontrar el algoritmo que acabamos de exponer y una serie de mejoras que introduce el autor, para el 1-FlowLoc, por medio de un preprocesamiento de los arcos de la red y las localizaciones factibles, consiguiendo unas diferencias de rendimiento respecto al algoritmo por enumeración de más de un 99% gracias, precisamente, a ese preprocesamiento. En estas mismas páginas podemos ver que el autor nos proporciona la complejidad computacional para resolver el problema del 1-FlowLoc mediante el Algoritmo 4 que es $\mathcal{O}(|\mathbb{L}||V|^3)$, para la definición de la notación $\mathcal{O}$ ver Anexo A.1.

En el caso de querer localizar $q > 1$ instalaciones tendremos que buscar los lugares en $\mathbb{L} \subset A$ tales que la reducción del flujo máximo original sea lo menor posible y a su vez no sobrepasar la restricción en el número de instalaciones por arco que nos da la función m.

Una primera tentativa es tomar el Algoritmo 4 e ir situando una a una todas las instalaciones en los arcos que afecten lo menos posible al valor del flujo máximo. Esta forma de proceder nos llevaría a recorrer todas las posibilidades de q instalaciones colocadas entre $|\mathbb{L}|$ localizaciones, esto es, un algoritmo del orden de $\binom{|\mathbb{L}|}{q}$ posibilidades, haciendo viable esta posibilidad solo para los casos en los que el número combinatorio sea pequeño, es decir, q o $\mathbb{L} - q$ sean pequeños. Hay que tener en cuenta que un algoritmo heurístico de este tipo sería muy similar a uno tipo *greedy*, puesto que iremos situando las instalaciones en sus mejores posiciones posibles de manera iterativa.

Sabemos que esto no siempre nos conduce a una solución óptima global, aunque sí factible, y en muchas ocasiones eficiente, en el sentido que es ϵ-óptima en valor para un $\epsilon > 0$ pequeño. Entonces, una de las primeras preguntas que nos formulamos es si el Problema q-FlowLoc se puede llegar a resolver en un tiempo aceptable, es decir, en tiempo polinomial, con un algoritmo exacto. La respuesta general es que no, ya que demostraremos que el problema es $\mathcal{NP}$-completo. Aunque también veremos una heurística refinada que funciona muy bien en la práctica y que consigue resultados factibles cercanos al óptimo en tiempos inferiores a los marcados por la tentativa greedy que hemos comentado antes.

Primero transformaremos el Problema q-FlowLoc a su problema análogo de decisión para poder afrontar la demostración de la $\mathcal{NP}$-completitud. Esto es posible hacer ya que todo problema de optimización tiene su contrapartida como problema de decisión. Como estamos maximizando un flujo, el problema de decisión será encontrar un flujo cuyo valor sea mayor o igual que algún $k \in \mathbb{Z}$, pero que no sea mayor que $k + 1$, es decir, el flujo es máximo y lo tenemos acotado por $k + 1$.

PROBLEMA DE DECISIÓN q-FLOWLOC

Input: Una red de flujo y localización $R_{\mathbb{P}} = ((V, A), (c, d, m))$.

Pregunta: ¿Existe una función de localización $l : \mathbb{P} \to \mathbb{L}$ tal que el valor del flujo máximo en R_l sea mayor o igual que un $k \in \mathbb{Z}$?

Recordemos que la red $R_l = (V, A, c_l)$ a la que se refiere el problema tiene por capacidades en los arcos la función $c_l(a) := c(a) - \max_{p \in \mathbb{P}} \{d(p) \ : \ l(p) = a\}$. Vamos a demostrar que este problema de decisión, equivalente al problema de optimización homónimo, es $\mathcal{NP}$-completo.

Teorema 1.4.3. *El problema de decisión de q-FlowLoc es $\mathcal{NP}$-completo.*

Demostración. Es obvio que el problema de decisión q-FlowLoc está en $\mathcal{NP}$ puesto que si la respuesta a la pregunta es "sí", esta puede ser comprobada en tiempo polinomial encontrando un flujo máximo en la red R_l con las capacidades modificadas por las instalaciones.

Ahora, mostraremos que el problema de decisión q-FlowLoc es reducible polinomialmente al problema 3-SAT, el cual sabemos que es $\mathcal{NP}$-completo (ver Teorema A.1.8). Consideremos $B = \{b_1, b_2, \ldots, b_m\}$ un conjunto de variables booleanas y C_i unas disyunciones de 3 literales, $C_i = c_{i,1} \vee c_{i,2} \vee c_{i,3}$ con $i = 1, \ldots, k$, donde cada literal $c_{i,j}$ es distinto, fijado el i, y representará a la variable booleana b_h o $\neg b_h$ para algún $h = 1, \ldots, m$ con "$\neg$" la negación lógica. A cada C_i se le denominará cláusula, consideremos la colección de cláusulas $C = C_1 \wedge C_2 \wedge \cdots \wedge C_k$, donde "$\wedge$" es la conjunción lógica.

La expresión booleana C tiene la estructura del problema 3-SAT (ver Anexo A.1) y a partir de ella se puede construir una red dirigida con $3 + k + 3m$ nodos y $1 + 4k + 5m$ arcos como la de la Figura 1.8, que es un ejemplo para $k = 4$ y $m = 5$, en tiempo polinomial. Veremos que esta red representará el problema 3-SAT en términos de un problema de flujo y localización y que, en ella, es equivalente resolver el problema de decisión q-FlowLoc que el problema 3-SAT.

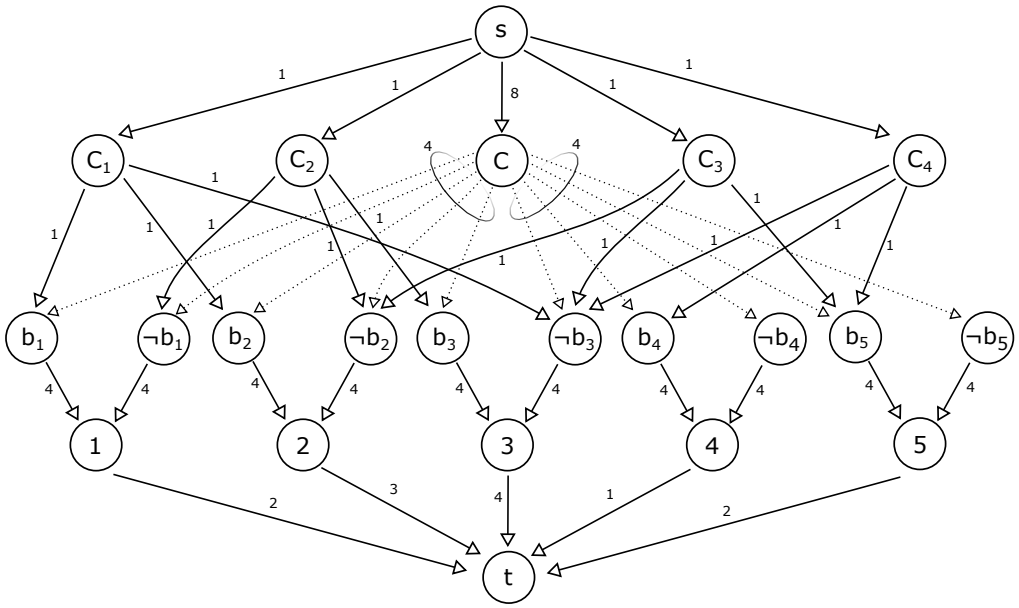

Figura 1.8: Red para representar el problema de decisión 3-SAT con cláusulas $C_1 = b_1 \vee b_2 \vee \neg b_3$, $C_2 = \neg b_1 \vee \neg b_2 \vee b_3$, $C_3 = \neg b_2 \vee \neg b_3 \vee b_5$ y $C_4 = \neg b_3 \vee b_4 \vee b_5$

Los nodos y arcos de la red R serán:

- Nodos fuente (s) y destino (t).
- Nodos claúsula C_i para cada $i \in \{1, \ldots, k\}$.
- Nodo colección C.
- Nodos literales b_h y $\neg b_h$ para cada $h \in \{1, \ldots, m\}$.
- Nodos de variables h para cada $h \in \{1, \ldots, m\}$.

- Arcos desde s hasta cada C_i para cada $i \in \{1, \ldots, k\}$ con capacidades 1.
- Arco desde s hasta C con capacidad $2k$.
- Arcos desde C_i para cada $i \in \{1, \ldots, k\}$ hasta sus literales $c_{i,j}$ con capacidades 1.
- Arcos desde C hasta todos los literales $c_{i,j}$ con capacidades k, en total $2m$ arcos.
- Arcos desde b_h y $\neg b_h$ hasta su variable h para cada $h \in \{1, \ldots, m\}$ con capacidades k.
- Arcos desde los nodos h para cada $h \in \{1, \ldots, m\}$ hasta t con capacidades igual a las veces que se repiten b_h y $\neg b_h$ en todas las clausulas C_i.

Solo los arcos que conectan los nodos literales b_h y $\neg b_h$ a los nodos de variables h para cada $h \in \{1, \ldots, m\}$ tienen capacidad para localizar una instalación por arco, el resto no.

Es obvio que la suma de las capacidades de los arcos que conectan el nodo s con sus sucesores es $3k$. Si construimos un flujo factible de valor $3k$ en la red, en virtud del Corolario 1.2.6, será flujo máximo. Supongamos que saturamos todos los arcos sucesores del nodo s, este flujo tendría valor $3k$, veamos que puede llegar hasta el nodo t. El flujo que atraviesa los arcos (s, C_i) $\forall i \in \{1, \ldots, k\}$ llega sin problema hasta el nodo t debido a que existen tres arcos sucesores (y que no inciden en el mismo literal) de cada nodo cláusula C_i.

El resto del flujo, con valor $2k$, atraviesa el arco (s, C) y existen arcos $(C, c_{i,j})$, con capacidad k cada uno, siendo el literal $c_{i,j} = b_h$ ó $c_{i,j} = \neg b_h$ para algún $h \in \{1, \ldots, m\}$. Entonces, dividimos el flujo siguiendo la siguiente lógica: Se hace pasar una unidad de flujo por el arco $(C, c_{i,j})$ por cada arco de la forma $(C_i, c_{i,j})$ que no haya sido utilizado ya por el flujo en el párrafo anterior. Puesto que no es un multigrafo (ver Figura 1.8), los arcos de la forma (b_h, h) y $(\neg b_h, h)$ nunca sobrepasan su capacidad, que es, k. De esta forma el flujo puede llegar hasta el nodo t, ya que las capacidades de los arcos (h, t) han sido fijadas según el número de arcos $(C_i, c_{i,j})$ que son predecesores de un literal $c_{i,j} = b_h$ ó $c_{i,j} = \neg b_h$ para cada $h \in \{1, \ldots, m\}$.

El problema q-FlowLoc en esta red se traduciría en ver si podemos localizar $q = m$ instalaciones de tamaño k tal que el flujo máximo de la red sea mayor o igual que $3k$. Veamos que la expresión booleana C del 3-SAT se satisface si y solo esto se cumple.

$\boxed{\Longrightarrow}$ Supongamos que la expresión C cumple el problema 3-SAT. Entonces cada cláusula C_i tiene, al menos, un literal $c_{i,j}$ que es verdad. Situando una instalación en el arco (b_h, h) si b_h es falsa o en el arco $(\neg b_h, h)$ si b_h es verdadera (para algún h) se tiene que existe un camino desde C_i hasta t para cada $i \in \{1, \ldots, k\}$ que envía un flujo de valor 1. Tenemos de momento un flujo con valor k que llega hasta t.

Como b_h no es falsa y verdadera simultáneamente, se tiene que en los arcos (b_h, h) y $(\neg b_h, h)$ no se sitúan instalaciones a la vez. Además, al nodo C llega un flujo de valor $2k$ y este se conecta con arcos de capacidad k a todos los literales a la vez, por tanto, es suficiente repartir este flujo de valor $2k$ entre todos los arcos disponibles de tal forma que a cada nodo de variable h llegue un flujo igual a la capacidad del arco (h, t) menos el flujo ya ocupado desde C_i hasta t descrito en el párrafo anterior.

Como la suma de las capacidades de los arcos incidentes en el nodo t es $3k$ el proceso que hemos descrito se puede realizar y tenemos que en la red R_l conseguimos un flujo con valor mayor o igual a $3k$.

$\boxed{\Longleftarrow}$ Las m instalaciones de tamaño k pueden colocarse en la red R de tal forma que el flujo máximo en la red R_l sea mayor o igual que $3k$.

Veamos que dos instalaciones no pueden localizarse en los arcos (b_h, h) y $(\neg b_h, h)$ simultáneamente para ningún $h \in \{1, \ldots, m\}$. Procedamos por reducción al absurdo, supongamos que $\exists h \in \{1, \ldots, m\}$ tal que dos instalaciones se localizan en los arcos (b_h, h) y $(\neg b_h, h)$, entonces, por la construcción de la red R estos arcos no tienen capacidad libre en R' pues las instalaciones tienen capacidad k, por tanto, el flujo en el arco (h, t) será cero y por ende, el máximo flujo será estrictamente menor que $3k$ puesto que las capacidades de los arcos que incidían en el nodo t eran exactamente $3k$ como hemos apuntado en un párrafo anterior.

Por tanto, C se satisface escogiendo como *verdaderas* las variables b_h si la capacidad del arco (b_h, h) es igual a k, es decir, se localiza una instalación en el arco $(\neg b_h, h)$ y *falsas* en el otro caso, en el que la instalación se localiza en el arco (b_h, h). □

Por supuesto, demostrar que en el peor de los casos no se conseguiría resolver el problema del q-FlowLoc para $q > 1$ en tiempo polinomial, a no ser que $\mathcal{P} = \mathcal{NP}$, no quiere decir que la comunidad científica deje de buscar algoritmos heurísticos que logren resolver la mayor parte de las instancias en tiempos aceptablemente rápidos.

En concreto, en Hamacher et al. (2013, p. 175), podemos encontrar un algoritmo heurístico que en el peor de los casos tiene un orden de $\mathcal{O}(q \log(q) + q(|\mathbb{L}| + n^3))$, con n el número de nodos, y que el propio autor nos compara con el tiempo que tarda el solver CPLEX en resolverlo de manera exacta, obteniendo resultados un 95% mejores, como mínimo, en tiempo. Siendo CPLEX incapaz de resolver alguna de las instancias presentadas en el artículo (ver Tabla 1.1) por su elevado coste computacional.

Nodos	$q = 100$		$q = 200$	
	Algoritmo 5	CPLEX	Algoritmo 5	CPLEX
100	0.009	0.94	0.018	5.52
300	0.231	10.357	0.437	57.63
500	0.760	41.127	1.304	N/A

Tabla 1.1: Comparación del tiempo, en segundos, de ejecución del Algoritmo 5 y CPLEX resolviendo (1.4), datos extraídos de Hamacher et al. (2013, p. 175)

La heurística la presentamos a continuación con la notación adaptada a la que hemos dado en este capítulo. Esta se basa en ordenar las instalaciones por orden decreciente, buscar el arco a con mayor capacidad residual $c(a) - x(a)$ siendo x un flujo máximo. En dicho arco, localizaremos tantas instalaciones como se pueda, esto nos lo indica la función $m(a)$. Si la instalación más grande situada en el arco a tiene un tamaño mayor que la capacidad residual $c(a) - x(a)$, entonces, esa instalación afectará al flujo máximo de la red y, por tanto, tendremos que recalcular un flujo máximo en la red teniendo en cuenta las instalaciones ya localizadas hasta ese momento.

Algoritmo 5 q-FlowLoc estático $\mathcal{O}(q\log(q) + q(|\mathbb{L}| + n^3))$

Input: Red de flujo y localización $R_{\mathbb{P}} = ((V, A), (c, d, m))$. Siendo $\mathbb{P}$ el conjunto de instalaciones.

Pasos:

1. Definimos $k = 1$. Colocamos $\mathbb{P} = \{p_1, \ldots, p_q\}$ en orden decreciente según su tamaño
$$d(p_1) \geq d(p_2) \geq \cdots \geq d(p_q).$$
2. Calculamos flujo máximo x en la red R con el Algoritmo 2.
3. Si $k > q$, saltamos al paso 7. Si no, buscamos arco $\hat{a} \in \arg\max\{c(a) - x(a) : a \in \mathbb{L}\}$.
4. Fijamos la localización $l(p_{k+i-1}) = \hat{a} \;\; \forall i = 1, \ldots, \min\{m(\hat{a}), q + 1 - k\}$.
5. Actualizamos $c(\hat{a}) = \max\{c(\hat{a}) - d(p_k), 0\}$, $k = k + \min\{m(\hat{a}), q + 1 - k\}$, $\mathbb{L} = \mathbb{L}\backslash\{\hat{a}\}$.
6. Si $x(\hat{a}) > c(\hat{a})$ entonces volvemos al paso 2. Si no, volvemos al paso 3.
7. Finalizamos el proceso con flujo máximo x, valor $v(x)$ y localizaciones l. FIN.

Esta heurística ordena q elementos en un vector que tiene un coste en tiempo computacional de $\mathcal{O}(q\log(q))$. Además en cada iteración k calcula, a lo sumo, un flujo máximo con $\mathcal{O}(n^3)$, donde n es el número de nodos, y un arg max con una complejidad igual a una búsqueda en la una lista, es decir, $\mathcal{O}(|\mathbb{L}|)$. Quedando una complejidad computacional total de $\mathcal{O}(q\log(q) + q(|\mathbb{L}| + n^3))$.

Si nos fijamos, la clave del funcionamiento de esta heurística es precisamente que hemos supuesto que solo la instalación de mayor tamaño condiciona el flujo que pasa por un arco (véase Figura 1.6) y por ello en el paso 5 actualizamos la capacidad $c(\hat{a})$ disminuyendo en una cantidad igual al tamaño de la mayor instalación colocada en esa iteración, que es la instalación p_k. Si tomásemos el modelo en el que las instalaciones afectan a la capacidad del arco de manera conjunta (como con la suma, véase Figura 1.7), entonces el Algoritmo 5 fallaría tal y como está expresado y habría que añadir otros pasos para asegurar que la solución final que se encuentra fuera, al menos, factible.

2. Flujos dinámicos

En la primera sección fijaremos el modelo que utilizaremos en la teoría del resto de secciones. Es posible que la necesidad de un modelo muy aproximado a la realidad haga que este se vuelva muy complejo de manipular matemáticamente. En ese sentido, la labor del matemático debe ser encontrar el equilibrio entre una buena explicación del suceso real por medio del modelo y una complejidad tratable para ser desarrollado tanto teórica como de manera práctica. Puesto que nosotros intentamos fijar un marco para un modelo de evacuación, podemos suponer, por ejemplo, que el proceso de evacuación se realiza en fases y discretizar así el tiempo.

Definiremos el concepto de *flujo dinámico factible*, resaltando su parecido con un flujo estático en el que consideramos la componente temporal como otra dimensión sobre la que trabajar y no un mero coste, y la *red expandida temporalmente*, la cual nos ofrecerá una biyección entre los problemas estáticos y los dinámicos. Al igual que en el capítulo anterior, plantearemos los problemas más relevantes que hay en la literatura. Estos son el problema del *flujo dinámico máximo*, el problema del *flujo máximo universal* y el problema del *flujo más rápido*. Explicaremos diferentes algoritmos para la resolución de cada uno de ellos y mostraremos la interrelación que existe entre ellos.

Para afrontar el problema de un flujo máximo dinámico nos basaremos en las ideas de Ford Jr y Fulkerson (1958). Definiremos lo que es una *descomposición por caminos* de un flujo y construiremos, dado un flujo factible estático, el *flujo repetido temporalmente* (o TRF). Se verá, en el Teorema 2.2.7, que es suficiente buscar un flujo de esta clase para maximizar el valor del flujo dinámico en un horizonte temporal y que, para ello, podremos resolver el problema del flujo estático a coste mínimo en una red modificada con acierto. A continuación, ampliaremos el concepto de *corte de una red* al caso dinámico y mostraremos un ejemplo explícito.

Un flujo máximo universal es un flujo máximo dinámico (ver Definición 2.3.9) pero, en lugar de para un solo horizonte temporal fijo, para una serie de horizontes simultáneamente. Dentro del flujo máximo universal distinguiremos el flujo de llegada y el flujo de salida, dando lugar a los cuatro subproblemas del *flujo de llegada/salida más temprana/tardía*. Será en ese contexto, con el Teorema 2.4.10, en donde veremos que un flujo de llegada más temprana también genera un flujo más rápido en la red R. Aunque esta no es la forma eficiente de plantear el problema del flujo más rápido, pues veremos los Algoritmos 8 y 9, que resuelven el problema en tiempo polinomial y mediante unos razonamientos más propios del análisis matemático.

Terminaremos el capítulo con la generalización del problema de flujo y localización al caso dinámico y que, gracias al Teorema 2.2.7, se plantea igual que el caso estático pero encontrando un flujo máximo a coste mínimo en la red que se vea interrumpido lo menos posible por las instalaciones que se quieren colocar.

2.1. Especificaciones del modelo

Al hablar de flujos dinámicos debemos fijar un modelo para la componente temporal. Este puede ser continuo o discreto. Nosotros desarrollaremos este último, pero al lector que quiera profundizar en una generalización en tiempo continuo, en la que se hace uso de integrales para calcular el flujo, se le recomienda leer Skutella (2009). Fijaremos un horizonte temporal $H \in \mathbb{N}$ finito y distribuiremos los flujos que pasan por la red en tiempo discreto $\theta = 0, 1, 2, \ldots, H$.

Al tener en cuenta el tiempo dentro del proceso de modelización el concepto de red que vimos en la Definición 1.1.2 no contempla todas las posibilidades que se pueden dar. Pensemos en una red que modela una planta de un edificio. Las habitaciones serían los nodos y los pasillos que las conectan los arcos. Es evidente que, si queremos captar el hecho de que la gente se quede en una habitación a lo largo del tiempo, no podemos cuantificarlo con una red como habíamos visto hasta ahora y tendremos que añadir una capacidad a los nodos, en nuestro ejemplo, un máximo número de personas que pueden estar en una habitación.

Definición 2.1.1. Sea $R = ((V, A), (c, \tau))$ una red dirigida con capacidades en los arcos c y una función de costes de tránsito τ. Si extendemos la definición de la función de capacidad a los nodos de manera natural $c : A \cup V \longrightarrow \mathbb{R}^+$ entonces R será una *red de espera* dirigida.

Fijémonos en que la restricción de c al conjunto de arcos $c|_A$ es la función de capacidades original. Esta definición es más bien técnica, pues podríamos haber definido una red dirigida en un principio con capacidades en los nodos y que dicha característica la usáramos en la teoría a partir de ahora. Por tanto, en este capítulo, cuando hablemos de red supondremos implícitamente que es una red de espera y que la función de capacidades está definida en los nodos.

Por otro lado será conveniente, en los casos prácticos, que se cuantifique el número de individuos que están en un nodo en cierto momento. Esto vendrá representado matemáticamente por el flujo que pasa por unos arcos, que denominaremos arcos de espera, de una transformación de la red. Dicha transformación es la que llamaremos red expandida temporalmente hasta un cierto horizonte.

Definición 2.1.2. Sea $R = ((V, A), (c, \tau))$ una red y fijemos H un horizonte temporal. Entonces la *red expandida temporalmente* es la red $R_H := ((V_H, A_H), (c_H, \tau_H))$ donde:

$$V_H = \{i_\theta : i \in V;\ \theta = 0, \ldots, H\} \quad \text{(Copias temporales de los nodos)},$$

$$A_H = \{(i_\theta, i_{\theta+1}) : i \in V;\ \theta \leq H - 1\} \cup \{(i_\theta, j_{\theta+\tau(i,j)}) : (i, j) \in A;\ \theta \leq H - \tau(i, j)\}.$$

Sus capacidades y costes de tránsito son:

$$c_H(i_\theta, i_{\theta+1}) = c(i), \quad c_H(i_\theta, j_{\theta+\tau(i,j)}) = c(i, j),$$

$$\tau_H(i_\theta, i_{\theta+1}) = 1, \quad \tau_H(i_\theta, j_{\theta+\tau(i,j)}) = \tau(i, j).$$

Observación. Fijémonos que una red expandida temporalmente es una red dirigida en el sentido estático en la cual hemos definido unos *arcos de espera* entre las copias de los nodos. En la Figura 2.1 estos arcos son los discontinuos. El flujo que pase por estos arcos $x(i_\theta, i_{\theta+1})$ representará a los individuos que queden esperando en un nodo durante un lapso de tiempo.

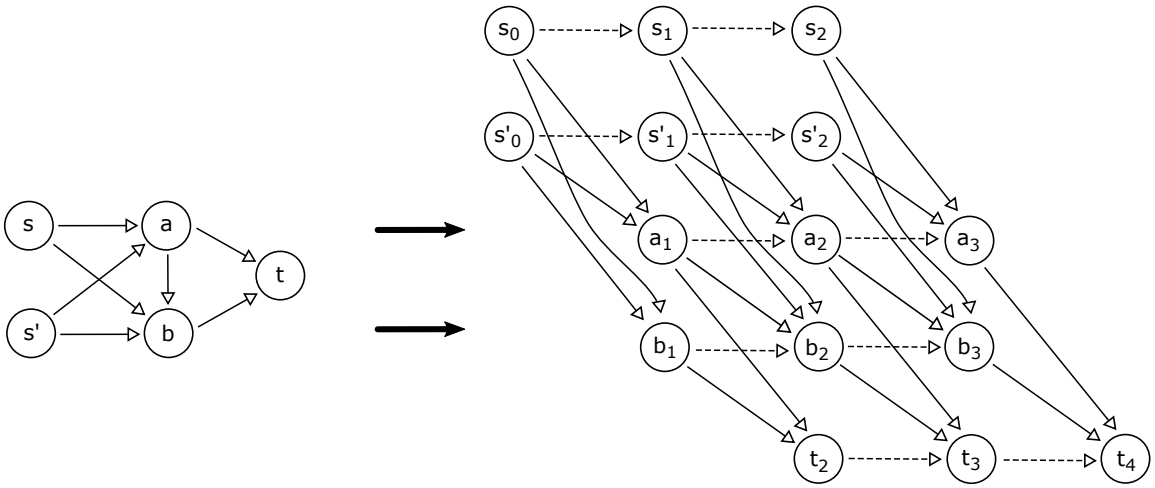

Figura 2.1: Red con $\tau \equiv 1$ en todo arco y su expansión temporal en horizonte $H = 4$

Como vemos en nuestro ejemplo, lo habitual es que se generen copias de nuestros nodos fuente y destino. En la Figura 2.1 hemos querido incluir dos nodos fuente, s y s', para hacer notar al lector que los casos de múltiples orígenes iniciales son posibles y que no es necesario crear un "súper origen" en la red inicial, tal y como dijimos en el Capítulo 1, sino que es suficiente con hacerlo en la red expandida como veremos en la Figura 2.2.

Estos casos son tremendamente típicos en evacuación, pues si se pretende evacuar una planta de un edificio con varias habitaciones no podemos modelizarlo simplemente suponiendo que los individuos en cada una de ellas puedan comenzar la evacuación/flujo desde un nodo madre hacia cualquier otro nodo fuente, sino que tendremos que representar la posibilidad de varios nodos fuente y limitar el flujo de alguna manera.

Esta limitación la lograremos en la red expandida temporalmente creando nodos fuente y destino s y t que aglutinen al resto de nodos s_θ, s'_θ, t_θ $\forall \theta \leq H$ por medio de unos arcos de manera coherente con el modelo real subyacente.

Estos arcos (s, s_θ), (s, s'_θ), (t_θ, t) $\forall \theta \leq H$ tendrán unas capacidades y costes asociados. Veamos un razonamiento para asignarlos. Puesto que en la red expandida temporalmente habrá siempre arcos de espera entre las copias de un mismo nodo, entonces no es necesario definir los arcos (s, s_θ) y (s, s'_θ) $\forall \theta > 0$, porque si queremos que un flujo comience desde, por ejemplo, el nodo s_1, entonces dicho flujo deberá pasar por (s, s_0) y después por el arco de espera (s_0, s_1). Sin embargo, la decisión de definirlos explícitamente o no, depende del contexto del problema y las técnicas de resolución que se lleven a cabo en cada caso.

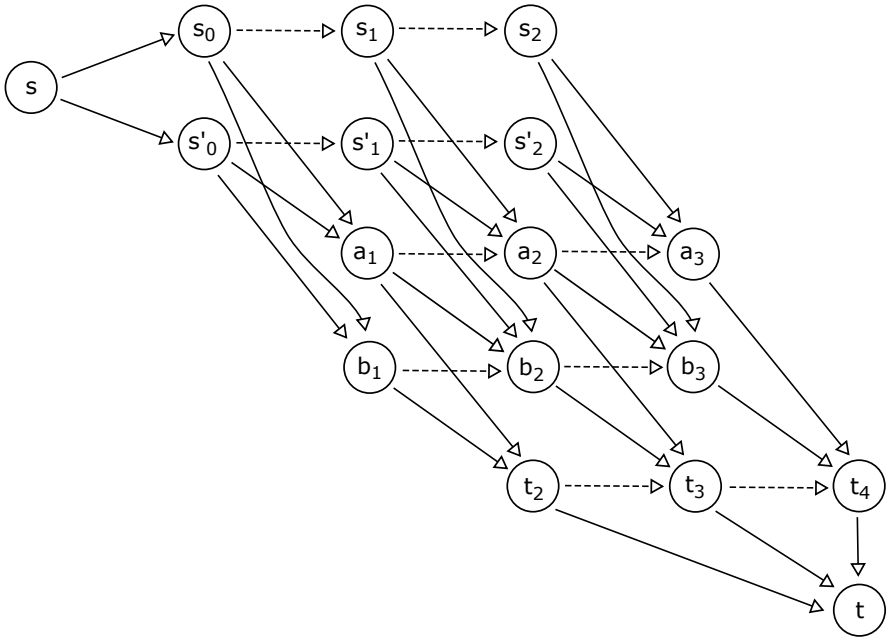

Figura 2.2: Red expandida temporalmente con nodos fuente s y destino t comunes

Los únicos arcos absolutamente necesarios desde el nodo fuente s serán hacia las copias en tiempo 0 de todas las fuentes que hubiese en la red original. Sus costes temporales serán nulos, ya que es una representación matemática no real, y sus capacidades serán iguales a las capacidades de los nodos fuente de la red o, si se conociese, la cantidad de individuos en su interior en un momento dado. Es decir,

$$c(s, s_0) = c(s),\ \tau(s, s_0) = 0,$$

$$c(s, s'_0) = c(s'),\ \tau(s, s'_0) = 0,$$

Hagamos notar al lector que en la Figura 2.1 solo hay un nodo destino en la red, pero de forma muy similar se puede razonar la posibilidad de varios nodos destino como salidas separadas físicamente. Sin embargo, en este caso, sí vamos a considerar todos los arcos (t_θ, t) $\forall \theta \leq H$ y si hubiese más nodos destino, todos los asociados a estos.

Las capacidades de los arcos (t_θ, t) deben ser, por lo menos, tan grandes como para soportar el máximo flujo posible, es decir, que al menos tienen que tener la capacidad de los nodos t_θ. En la práctica se les asigna un valor infinito. En el caso de los costes de tránsito, diremos que el coste de pasar por un arco (t_θ, t) es igual al tiempo en el que se llega al arco, es decir, θ.

$$c(t_\theta, t) = \infty,\ \tau(t_\theta, t) = \theta \quad \forall \theta \leq H.$$

Estos costes temporales τ en los arcos (t_θ, t) se denominan en la literatura (ver Chalmet et al., 1982, p. 97) "turnstile costs". Su importancia radica en el hecho de que gracias a ellos se puede definir un problema muy útil en flujos que es el del tiempo medio de salida del flujo de una red (ver la observación del Teorema 2.4.10).

Una vez hemos definido la red, que será nuestra estructura de trabajo, vamos a definir nuestro instrumento de estudio, el flujo. La definición de flujo factible estático (ver Definición 1.2.1) puede ser ampliada para modelizar un **flujo factible dinámico** que varíe según el momento temporal en el que se encuentra el flujo pasando por cada arco.

Definición 2.1.3. Dada una red $R = ((V, A), (c, \tau))$, siendo c la capacidad de cada arco o nodo y τ el tiempo de tránsito por un arco. Diremos que $x : A \times \{0, 1, \dots, H\} \to \mathbb{R}^+$ es un *flujo factible dinámico* de s a t con horizonte temporal H si cumple las siguientes condiciones:

- No negatividad: $x(a, \theta) \geq 0, \forall a \in A, \theta \in \{0, 1, \dots, H\}$.
- Restricción de capacidad: $x(a, \theta) \leq c(a), \forall a \in A, \theta \in \{0, 1, \dots, H\}$.
- Cuasi-conservación de flujo: $\forall i \in V \backslash \{s, t\}, \theta \in \{0, 1, \dots, H\}$

$$\sum_{a \in \Gamma^{-1}(i)} \sum_{z=\tau(a)}^{\theta} x(a, z - \tau(a)) \geq \sum_{a \in \Gamma(i)} \sum_{z=0}^{\theta} x(a, z).$$

- Capacidad nodal: $\forall i \in V \backslash \{s, t\}, \theta \in \{0, 1, \dots, H\}$

$$\sum_{a \in \Gamma^{-1}(i)} \sum_{z=\tau(a)}^{\theta} x(a, z - \tau(a)) - \sum_{a \in \Gamma(i)} \sum_{z=0}^{\theta} x(a, z) \leq c(i).$$

- Conservación de flujo en H: $\forall i \in V \backslash \{s, t\}$

$$\sum_{a \in \Gamma^{-1}(i)} \sum_{z=\tau(a)}^{H} x(a, z - \tau(a)) = \sum_{a \in \Gamma(i)} \sum_{z=0}^{H} x(a, z).$$

Las condiciones de no negatividad y restricción de capacidad son obvias para mantener la lógica del modelo en la red R para todo tiempo θ. La condición de cuasi-conservación de flujo se puede interpretar como que un flujo dinámico puede dejar esperando parte de su flujo en un nodo hasta cuando sea necesario dentro del horizonte temporal H y, por lo tanto, el flujo que entra siempre es mayor o igual que el que sale, excepto en tiempo H, ya que en ese momento sí debe existir conservación de flujo en toda la red. La condición de capacidad nodal nos limita la cantidad de flujo que puede quedarse en espera en un nodo por su capacidad en la red original, algo que tiene sentido al querer modelizar espacios físicos reales no infinitos.

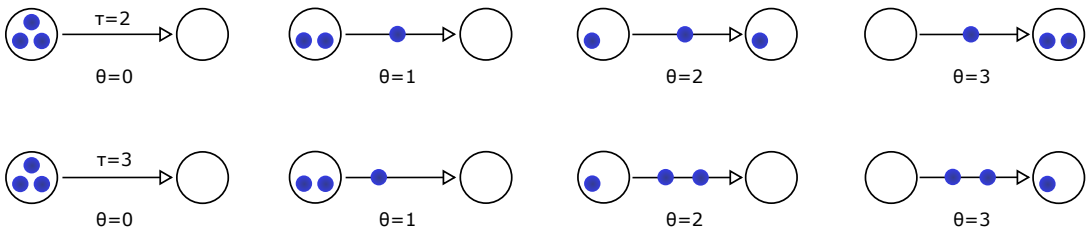

Figura 2.3: Representación discreta de un flujo dinámico entre dos nodos para $H = 3$ y dos tiempos de tránsito $\tau = 2$ y $\tau = 3$

En la Figura 2.3 se ha representado el flujo mediante círculos, con toda la capacidad disponible $c(a)$, que se mueven a lo largo del arco en cada instante θ de nuestra discretización. Como en el primer caso el arco tiene un tiempo de tránsito de $\tau = 2$, el flujo que se transmite hasta tiempo $H = 3$ es de $2c(a)$, dos círculos. En el segundo caso, puesto que $\tau = 3$, tenemos que el flujo transmitido solo es de un círculo con valor $c(a)$. De forma análoga al caso estático, definiremos el *valor de un flujo dinámico* de s a t como la cantidad de flujo que llega a t y que, por conservación de flujo en el horizonte H, será:

$$v_H(x) := \sum_{a \in \Gamma^{-1}(t)} \sum_{z=H_0}^{H} x(a, z - \tau(a)) = \sum_{a \in \Gamma(s)} \sum_{z=0}^{H-H_0} x(a, z). \tag{2.1}$$

Observación. Posteriormente escribiremos de manera formal el término H_0, pero de momento es suficiente con saber que es el tiempo del camino más corto (estático) desde s hasta t en la red R original. Vemos que en el segundo sumatorio de la primera igualdad el flujo que se considera empieza a tomar valor en $H_0 - \tau(a)$, porque si en el instante $H_0 - \tau(a)$ el flujo no ha llegado a un predecesor de t, entonces en un instante inferior a H_0 no puede llegar, por su propia definición. Análogamente, termina de tomar valor en $H - \tau(a)$, ya que si en el instante $H - \tau(a)$ el flujo no ha podido atravesar el arco $a \in \Gamma^{-1}(t)$, entonces en el instante H no llegará al nodo de salida t y, por tanto, no podemos considerarlo parte del flujo hasta ese horizonte temporal (ver Figura 2.3). El caso del segundo sumatorio de la segunda igualdad es similar. Mediante un razonamiento análogo al anterior, si en un instante superior a $H - H_0$ hay un flujo que sale del nodo s, este no podrá alcanzar el nodo t en el horizonte temporal H por la definición de H_0 como el tiempo del camino más corto.

Observación. El valor H_0 definido antes se puede calcular fácilmente con, por ejemplo, el Algoritmo de Dijkstra (ver Dijkstra, 1959). Y se puede definir matemáticamente de la siguiente forma: sea $\mathcal{P}$ el conjunto de todos los caminos en la red R desde el nodo fuente s al destino t y $\tau(P) = \sum_{a \in P} \tau(a)$, entonces $H_0 := \min\{\tau(P) : P \in \mathcal{P}\}$.

Veamos la relación entre flujo estático en la red expandida y flujo dinámico. Si llamamos x_H a dicho flujo en la red expandida y x al flujo dinámico en R, se tiene que:

$$x((i,j),\theta) = x_H\left(i_\theta, j_{\theta+\tau(i,j)}\right) \quad \forall (i,j) \in A,\ i,j \in V,\ \theta \in \{0,\dots,H\}.$$

Por la Definición 2.1.2 de red expandida, y que tiene en consideración las capacidades de los nodos en los arcos de espera, el flujo x_H respeta dichas capacidades y el flujo x también lo hará con la relación dada. Luego, si x_H es factible estático en la red expandida entonces x es factible dinámico en la red original.

Ejemplo 2.1.4.

Tomemos una red $R = ((V, A), (c, \tau))$ como la de la Figura 2.4 con los costes y capacidades en cada arco expresados como una dupla $(c(a), \tau(a))$. Entonces la red expandida temporalmente de R hasta tiempo $H = 5$ sería como la Figura 2.5, donde los arcos discontinuos son arcos de espera en un mismo nodo con capacidad la del nodo que unen, aunque no lo pongamos explícitamente en el dibujo por no saturarlo.

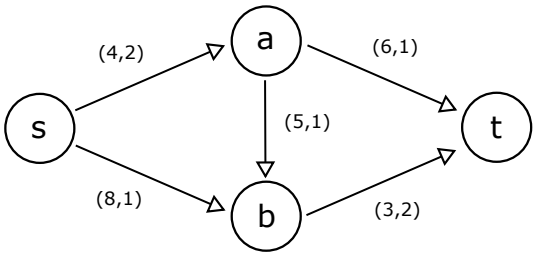

Figura 2.4: Red R con $(c(a), \tau(a))$

En la Figura 2.5 hemos construido un flujo factible x_H cualquiera que cumpla las condiciones de capacidad y conservación de flujo en la red expandida. Entonces, el flujo factible dinámico equivalente a este en la red R sería el de la sucesión de la Figura 2.6.

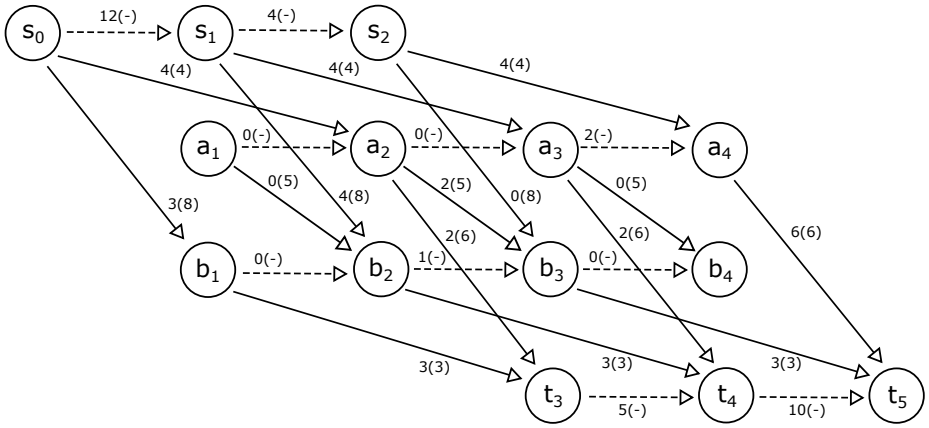

Figura 2.5: Red R expandida con horizonte temporal $H = 5$ y "$x_H(a)$ $(c(a))$" en cada arco

Con la notación utilizada se tiene $v_H(x) = v(x_H) = 19$ en este ejemplo. Sin embargo, no es el máximo flujo (ver Figura 2.8) que se puede tener en esta red de manera dinámica. El primer problema que plantearemos será este, encontrar un flujo máximo dinámico en una red R dada.

En el modelo que hemos fijado se pueden realizar innumerables ampliaciones y modificaciones como el hecho de pasar a tiempo continuo (ver Skutella, 2009). Por ejemplo, nosotros consideraremos un horizonte temporal H finito. Pero en algunos modelos, como los de tráfico, virtualmente el horizonte temporal es infinito y se modela con una ventana temporal hasta H fijo para posteriormente variarlo (ver Orlin, 1983).

En la red hemos supuesto que es posible esperar en los nodos ya que el modelo real de evacuación al que nos querríamos acercar sugiere que esto es lógico. Sin embargo, existen artículos como el de Fleischer y Skutella (2003), en los que se prohíbe expresamente las esperas en los nodos.

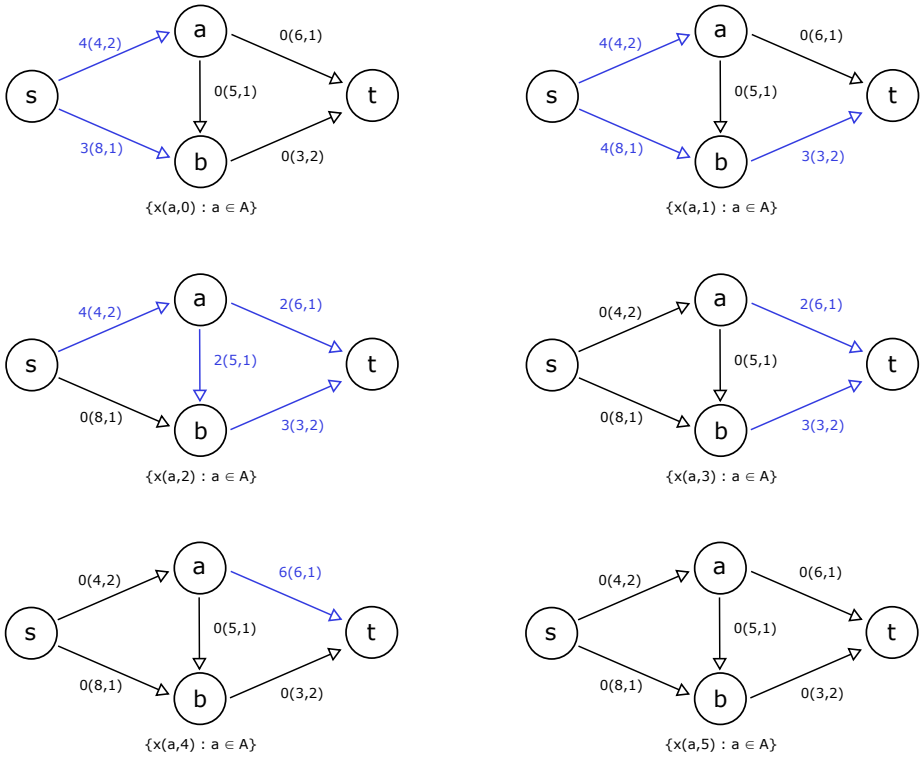

Figura 2.6: Representación del flujo dinámico x a través de R

El modelo puede obligar a considerar varios nodos fuente y destino totalmente diferenciados. Este modelo de red se denomina "multiterminal" y es usado, aparte de en problemas de evacuación con múltiples salidas y comienzos, en el problema del transporte dinámico, en el que en cada instante de tiempo hay que decidir qué flujo de bienes se envía desde una fuente específica hacia un destino concreto (ver Bookbinder y Sethi, 1980).

También se podrían variar parámetros a lo largo del tiempo, como las capacidades de los arcos mientras se transita por ellos (en una zona que esté parcialmente destruida, pasar por un camino puede bloquearlo) o los tiempos de tránsito (Orda y Rom, 1995). Añadir una variable ambiental estocástica dentro del proceso de decisión (ver una aplicación en rutas Azaron y Kianfar, 2003) conllevaría analizar la robustez de los métodos usados en el escenario medio y en el peor (ver aplicación a una evacuación Goerigk et al., 2018). E incluso el tiempo que se tarda en recorrer un arco podría depender del flujo que pase por él en un momento dado. Esto se puede ver en los casos de evacuación, en los que un flujo muy grande de personas puede saturar y ralentizar el movimiento del grupo total.

Por último, existen los flujos generalizados (ver Tardos y Wayne, 1998; Groß y Skutella, 2011) en los cuales tenemos la variable temporal, un tiempo de tránsito y una función de pérdida γ en cada arco, haciendo que el flujo que atraviesa un arco se reduzca en un tanto por ciento. Esto se usa en modelos reales de redes con posibles pérdidas (robos o deterioro) y en evacuación se podría usar para cuantificar el desgaste que está sufriendo una vía de escape al paso de la gente y cómo ese desgaste puede provocar que un arco quede con una capacidad efectiva nula.

2.2. Flujo máximo y corte dinámico

Al igual que hicimos en el caso estático, un primer problema que tenemos la tentación de plantear en una red con un flujo subyacente es el del flujo máximo. En este caso, fijado un horizonte temporal H, se darán condiciones para encontrar un flujo máximo dinámico en dicha red gracias a la red expandida temporalmente y al concepto, que luego desarrollaremos, de flujo repetido temporalmente (ver Definición 2.2.3).

Ejemplo 2.2.1. Un ejemplo inmediato en el que es necesario usar un flujo dinámico se tiene al intentar modelizar la máxima cantidad de personas que pueden bajar por unas escaleras en un tiempo fijo. Es obvio que las escaleras tienen una capacidad física de personas pero, una vez el primer grupo ha comenzado a bajar, puede bajar otro grupo de personas del mismo tamaño que el original pero en tiempo distinto. Esto es lo que se representaba mediante la Figura 2.3 con un nodo inicio y otro final. Los círculos serían los grupos de personas bajando en cada instante de tiempo θ.

De manera formal presentaremos el problema del flujo máximo como sigue:

Flujo máximo dinámico

Input: Sea $R = ((V, A), (c, \tau))$ una red, s y t los nodos fuente y destino, H un horizonte temporal.

Pregunta: Maximizar $v_H(x)$.

Dentro del contexto de las estrategias de evacuación, el problema del flujo máximo dinámico respondería a la pregunta de cuántas personas, a lo sumo, se pueden evacuar desde el punto s al punto t de forma segura en un horizonte temporal H.

Una primera aproximación para resolver el problema del flujo dinámico máximo es intentar reducir al caso anterior, es decir, dada una red dirigida $R = ((V, A), (c, \tau))$, ¿es posible construir otra red R', sin la componente temporal, cuyo flujo máximo sea equivalente a un flujo dinámico en R? La respuesta es afirmativa y esto se puede hacer con la red expandida temporalmente $R' = R_H$ (ver Definición 2.1.2), reduciendo el problema de encontrar un flujo máximo dinámico de s a t a encontrar un flujo máximo estático en R_H desde s_0 hasta t_H.

La segunda manera de encontrar un flujo máximo dinámico se la debemos a Ford Jr y Fulkerson (1958) (ver también Ford Jr y Fulkerson, 1962, p. 142-151) y es el llamado *flujo repetido temporalmente* o TRF por sus siglas en inglés. Se basa en que, dado un flujo estático factible, este se puede descomponer por caminos en un conjunto de flujos factibles. Cada camino se recorre tantas veces como sea posible hasta el horizonte H. Para entender esto primero vamos a definir lo que es un *camino de descomposición* de un flujo dado en una red.

Definición 2.2.2. Sea $R = (V, A, c)$ una red y x un flujo factible estático (ver Definición 1.2.1). Consideremos $\mathcal{P}$ el conjunto de todos los caminos en la red R desde el nodo fuente s al destino t y un conjunto de flujos factibles $\{x_P\}_{P\in\mathcal{P}}$ cumpliendo la siguiente propiedad:

$$x(a) = \sum_{P\in\mathcal{P}} x_P(a) \qquad \forall a \in A.$$

Al conjunto $\{x_P\}_{P\in\mathcal{P}}$ lo llamaremos descomposición del flujo x por caminos.

Observación. Sabemos que la descomposición por caminos existe siempre gracias a la *rutina II* que aparece en Ford Jr y Fulkerson (1958, p.422-423). Por tanto, la suma de la definición anterior está bien definida sabiendo que $x_P(a) = 0$ si $a \notin P$.

Definición 2.2.3. Sea $R = ((V, A), (c, \tau))$ una red, x un flujo factible estático y $\{x_P\}$ una descomposición del flujo x para los caminos $P \in \mathcal{P}$. El **flujo repetido temporalmente** (TRF), que denotaremos por $\bar{x}$, envía $v(x_P)$ unidades de flujo a través del camino P en cada periodo $0, 1, \ldots, H - \tau(P)$, donde $\tau(P) = \sum_{a\in P} \tau(a)$ es el coste total en tiempo del camino P. Por tanto, el valor del TRF será

$$v_H(\bar{x}) = \sum_{P\in\mathcal{P}} (H - \tau(P) + 1)v(x_P).$$

Observación. Como vemos, lo que hace un TRF es repetir el flujo que se envía por el camino P tantas veces como sea posible, hasta el horizonte temporal H, según el coste temporal del camino P, que es $\tau(P)$.

Ejemplo 2.2.4. Vamos a calcular el valor del TRF $\bar{x}$ dado un flujo máximo en la Figura 2.4 y veremos que $v(\bar{x})$ coincide con el valor del flujo máximo estático calculado en la red expandida de la Figura 2.5.

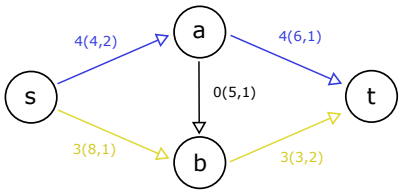

Figura 2.7: Red R con su descomposición en dos caminos

En la Figura 2.7 podemos ver un flujo máximo estático con una descomposición en dos caminos, uno amarillo y otro azul. Teniendo en cuenta la función de coste $\tau(a)$ en cada arco y que $H = 5$ tenemos que:

$$P_1 = \{(s, a), (a, t)\}, \qquad v(x_{P_1}) = 4, \qquad \tau(P_1) = 3,$$

$$P_2 = \{(s, b), (b, t)\}, \qquad v(x_{P_2}) = 3, \qquad \tau(P_2) = 3.$$

Por tanto, el valor del flujo repetido temporalmente será:

$$v_H(\bar{x}) = (H - \tau(P_1) + 1)v(x_{P_1}) + (H - \tau(P_2) + 1)v(x_{P_2}) = 21.$$

Si tomamos este flujo $\bar{x}$ en la red expandida temporalmente de la Figura 2.8 vemos que se puede encontrar un corte $\langle S, T\rangle$ en rojo con capacidad 21, por el Corolario 1.2.6 se tiene que este TRF $\bar{x}$ es un flujo máximo estático en la red expandida y, por tanto, podemos definir un flujo máximo dinámico en la red original a partir de él.

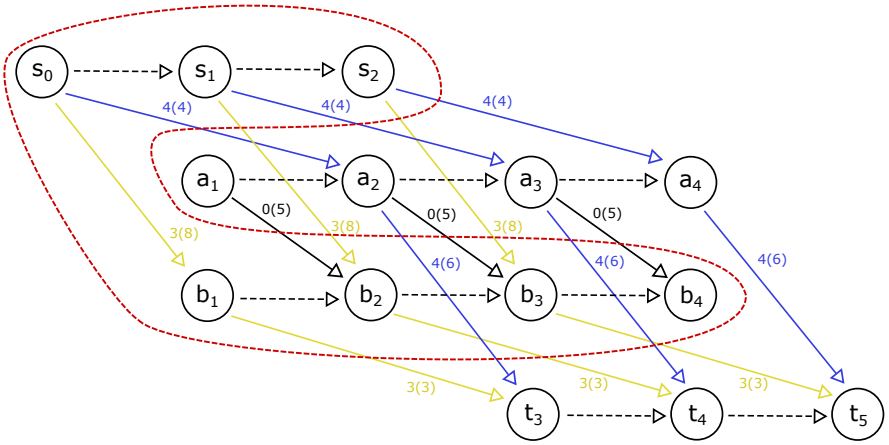

Figura 2.8: Red R expandida con horizonte temporal $H = 4$ y corte $\langle S, T\rangle$

El Ejemplo 2.2.4 da pie a preguntarnos cuándo podemos tener un flujo máximo dinámico a partir de un flujo estático. Para verlo vamos a demostrar un par de lemas previos que son propiedades técnicas de la descomposición de un flujo por caminos.

Lema 2.2.5. *Sea $R = (V, A, c)$ una red, x un flujo factible estático de $s \in V$ a $t \in V$ y $\{x_P\}$ una descomposición por caminos del flujo x, entonces:*

$$v(x) = \sum_{P \in \mathcal{P}} v(x_P).$$

Demostración. Por la definición de valor de un flujo estático dada en la sección anterior y la Definición 2.2.2 de descomposición por caminos de un flujo:

$$v(x) = \sum_{a \in \Gamma(s)} x(a) = \sum_{a \in \Gamma(s)} \sum_{P \in \mathcal{P}} x_P(a) = \sum_{P \in \mathcal{P}} \sum_{a \in \Gamma(s)} x_P(a) = \sum_{P \in \mathcal{P}} v(x_P).$$

□

Lema 2.2.6. *Sea $R = (V, A, c)$ una red, x un flujo factible estático de $s \in V$ a $t \in V$ y $\{x_P\}$ una descomposición por caminos del flujo x, entonces:*

$$x(a) = \sum_{P : a \in P} v(x_P) \quad \forall a \in A.$$

Demostración. Fijemos $a \in A$. Por la Definición 2.2.2 de descomposición por caminos y que $x_P(a) = 0$ si $a \notin P$ se sigue

$$x(a) = \sum_{P \in \mathcal{P}} x_P(a) = \sum_{P: a \in P} x_P(a).$$

Ahora fijémonos en la otra parte de la igualdad. Consideremos un corte de R cualquiera $\langle S, T \rangle$. Sabemos, por la Proposición 1.2.4, que el valor de un flujo es igual al flujo a lo largo de cualquier corte sobre la red:

$$\sum_{P: a \in P} v(x_P) = \sum_{P: a \in P} \left(\sum_{b \in \langle S,T \rangle^+} x_P(b) - \sum_{b \in \langle S,T \rangle^-} x_P(b) \right).$$

En particular, al ser P un camino, existe un corte $\langle S, T \rangle$ tal que el único arco de P que está en $\langle S, T \rangle^+$ es el arco a. Puesto que $x_P(a) = 0$ si $a \notin P$, eso quiere decir que el único término no nulo del sumatorio anterior es $x_P(a)$ con $a \in P$, quedando:

$$\sum_{P: a \in P} v(x_P) = \sum_{P: a \in P} x_P(a).$$

□

Ahora podemos demostrar que, a partir de un flujo factible, tendremos un flujo factible dinámico *en la familia de los TRF* y en las observaciones veremos cómo construirlo de manera que sea máximo.

Teorema 2.2.7. *Dada una red $R = ((V, A), (c, \tau))$ y x un flujo factible estático. El valor de un flujo factible dinámico $\bar{x}$ en la familia de los TRF para un horizonte temporal H viene dado por:*

$$v_H(\bar{x}) = (H + 1)v(x) - \sum_{a \in A} \tau(a)x(a).$$

Demostración. Sea $\{x_P\}_{P \in \mathcal{P}}$ una descomposición por caminos de x cualquiera. Definimos el TRF $\bar{x}$ a partir de esa descomposición. Entonces:

$$v_H(\bar{x}) = \sum_{P \in \mathcal{P}} (H - \tau(P) + 1)v(x_P) = (H + 1) \sum_{P \in \mathcal{P}} v(x_P) - \sum_{P \in \mathcal{P}} \tau(P)v(x_P).$$

Por la definición de $\tau(P)$ lo anterior es igual a:

$$(H + 1) \sum_{P \in \mathcal{P}} v(x_P) - \sum_{P \in \mathcal{P}} \sum_{a \in P} \tau(a)v(x_P).$$

Reordenando términos del segundo sumatorio se tiene:

$$v_H(\bar{x}) = (H + 1) \sum_{P \in \mathcal{P}} v(x_P) - \sum_{a \in A} \tau(a) \sum_{P: a \in P} v(x_P).$$

Aplicando los dos lemas anteriores se obtiene directamente el resultado del enunciado. □

Observación. Es flujo factible dinámico de forma obvia por construcción a partir de la descomposición por caminos de un flujo factible estático. Además, la expresión del valor de un TRF no depende de la descomposición por caminos del flujo estático inicial.

Observación. Si tomamos la red $R = ((V, A), (c, \tau))$ y le añadimos un arco adicional del nodo t a s con coste $\tau(t, s) = -(H + 1)$ y capacidad $c(t, s) = \infty$, es obvio que al calcular un flujo a coste mínimo en esta red obtendremos la expresión

$$\min_x \left\{ \sum_{a \in A \cup (t,s)} \tau(a)x(a) \right\} = \min_x \left\{ -(H+1)v(x) + \sum_{a \in A} \tau(a)x(a) \right\}.$$

Puesto que, al querer minimizar el coste de un flujo x en esa red, se hará pasar el máximo flujo por el arco (t, s) con coste negativo, es decir, $x(t, s) = v(x)$. Obtendremos así un flujo máximo estático a coste mínimo.
Entonces, por el teorema anterior, estamos minimizando $-v_H(\bar{x})$ que es lo mismo que menos maximizar $v_H(\bar{x})$, por tanto, un *flujo máximo dinámico* (en la familia de los TRF) puede ser encontrado calculando un flujo estático a coste mínimo en la red modificada del párrafo anterior.

Algoritmo 6 Flujo máximo dinámico $\mathcal{O}(n^2 m)$

Input: Red $R = ((V, A), (c, \tau))$ y horizonte temporal H
Pasos:

1. Aplicamos el Algoritmo 3, junto con su observación, para encontrar un flujo máximo estático a coste mínimo en R.
2. Repetimos el flujo temporalmente hasta el horizonte H. El valor vendrá determinado por la fórmula del Teorema 2.2.7. FIN.

Ejemplo 2.2.8. Consideremos la red de la Figura 2.4, pero con $\tau \equiv 1$ y dos capacidades cambiadas, $c(a, t) = 3$ y $c(b, t) = 6$. Llamaremos a esta red R'. Sea $H = 4$ nuestro horizonte temporal en este caso. Como aparece en la Figura 2.9 izquierda, podemos calcular un flujo máximo estático en R' con $v(x) = 9$ y este se puede descomponer en 3 caminos como sigue:

$$P_1 = \{(s, a), (a, t)\}, \qquad v(x_{P_1}) = 3, \qquad \tau(P_1) = 2,$$

$$P_2 = \{(s, b), (b, t)\}, \qquad v(x_{P_2}) = 5, \qquad \tau(P_2) = 2,$$

$$P_3 = \{(s, a), (a, b), (b, t)\}, \qquad v(x_{P_3}) = 1, \qquad \tau(P_3) = 3.$$

Tenemos, por la Definición 2.2.3, que:

$$v_H(\bar{x}) = \sum_{P \in \mathcal{P}} (H - \tau(P) + 1)v(x_P) = 26.$$

Pero este flujo no es máximo en la red expandida, pues el flujo x no es a coste mínimo. Para verlo, vamos a construir un flujo y a coste mínimo en la red R'.

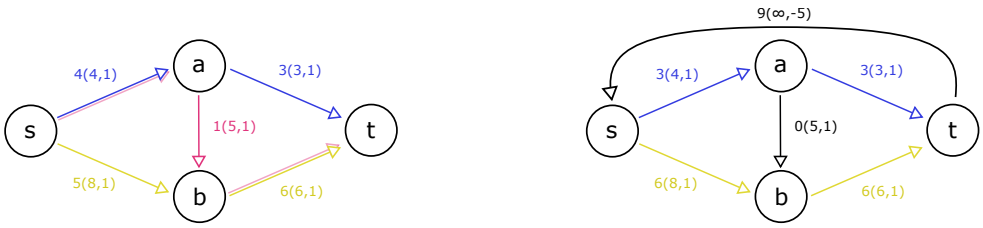

Figura 2.9: Red R' con su descomposición en tres caminos (izquierda) y red R' con el arco auxiliar (t, s) (derecha)

Como vemos en la Figura 2.9 derecha, añadimos el arco según la observación y aplicamos el algoritmo Busacker-Gowen (Algoritmo 3) para encontrar el flujo a coste mínimo con valor el mismo valor del flujo máximo anterior, es decir, $v(y) = v(x) = 9$.

Por el Teorema 2.2.7 tenemos que el valor de un TRF usando el flujo y será

$$v_H(\bar{y}) = (H+1)v(y) - \sum_{a \in A} \tau(a)y(a) = 27,$$

mayor que el valor del flujo $\bar{x}$ calculado antes. Además, por la observación, se tiene que este es un flujo máximo dinámico en la red R'. Esto se puede ver fácilmente en la red expandida de R' de la Figura 2.10 en la que encontramos un corte $\langle S, T \rangle$, en rojo, con capacidad el valor del flujo $\bar{y}$. Por el Corolario 1.2.6, el flujo $\bar{y}$ es un flujo máximo dinámico en R'.

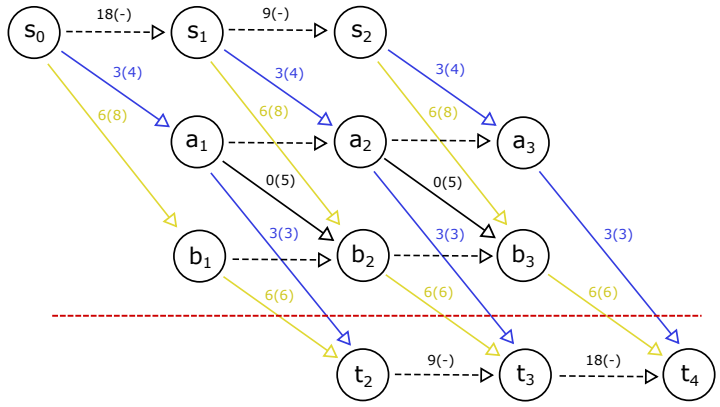

Figura 2.10: Red R' expandida con corte $\langle S, T \rangle$

Corte dinámico

Terminaremos la sección hablando de la generalización de corte en una red para un flujo dinámico, lo que diversos autores denominan *corte dinámico* o *corte a lo largo del tiempo* (ver Anderson et al., 1982). Lo haremos por construcción y empezaremos planteando el problema de programación lineal para obtener un flujo máximo:

Variables

x_a : flujo en el arco a $\quad \forall a \in A$.

Constantes

c_a : capacidad del arco $a \in A$.
τ_a : coste/tiempo del arco $a \in A$.
H : horizonte temporal.

Formulación

$$\text{max.} \quad (H+1)x_{(t,s)} - \sum_{a \in A \setminus \{(t,s)\}} \tau_a x_a \tag{2.2}$$

$$\text{s.a} \quad \sum_{a \in \Gamma^{-1}(i)} x_a - \sum_{a \in \Gamma(i)} x_a = 0 \qquad \forall i \in V \tag{2.2a}$$

$$0 \leq x_a \leq c_a \qquad \forall a \in A. \tag{2.2b}$$

En la segunda observación del Teorema 2.2.7 dedujimos que este problema era equivalente al de encontrar un flujo máximo dinámico, entre la familia de los TRF. Si nos fijamos, ha sido suficiente buscar un flujo estático y transformar la función objetivo desde un coste mínimo a la expresión del valor de flujo máximo del teorema. Si planteamos su dual, por medio de la tabla de Tucker, obtenemos:

Variables

α_i : provienen de las restricciones (2.2a) de conservación de flujo $\quad \forall i \in V$.
y_a : provienen de las restricciones (2.2b) de capacidad c_a $\quad \forall a \in A$.

Formulación

$$\text{min.} \quad \sum_{a \in A} c_a y_a \tag{2.3}$$

$$\text{s.a} \quad y_a + \alpha_i - \alpha_j \geq -\tau_a \qquad \forall a = (i,j) \in A \setminus \{(t,s)\} \tag{2.3a}$$

$$\alpha_t - \alpha_s \geq H + 1 \tag{2.3b}$$

$$y_a \geq 0 \qquad \forall a \in A. \tag{2.3c}$$

Se puede ver que la solución se alcanza con $y_a = \max\{0, \alpha_j - \alpha_i - \tau_a\}$ $\forall a = (i,j) \in A$, $\alpha_t \geq H + 1$ y $\alpha_s = 0$. A partir de este problema dual podemos definir, por construcción, lo que consideraremos la capacidad de un corte dinámico. Desearíamos que un corte dinámico cumpliese un teorema análogo al Teorema 1.2.11, en el que un corte de capacidad mínima definía a un flujo máximo estático. Pues, por el *Teorema de Dualidad Fuerte*, tenemos que el óptimo del problema (2.2) coincide en valor con el óptimo del problema (2.3), es decir, un flujo máximo dinámico tiene el mismo valor que minimizar la función objetivo de (2.3). Definimos ahora la capacidad de un corte dinámico.

Definición 2.2.9. Sea $R = ((V,A),(c,\tau))$ una red, H un horizonte temporal y $\alpha_i \in \mathbb{R}$ $\forall i \in V$ con $\alpha_s = 0$ y $\alpha_t \geq H + 1$. Entonces la capacidad de un corte dinámico que separa los nodos fuente s y destino t es

$$\sum_{a=(i,j) \in A} c_a \max\{0, \alpha_j - \alpha_i - \tau_a\}$$

Un corte de capacidad mínima tendrá el mismo valor que un flujo máximo dinámico por teoría de dualidad y hemos obtenido un teorema análogo al Teorema 1.2.11, pero para flujos dinámicos. Solo falta ver su interpretación geométrica, ya que un corte estático (ver Definición 1.2.2) empezaba nombrando una partición de los nodos de $V = S \cup T$ que dejaba a $s \in S$ y $t \in T$ separados.

Tomamos $\alpha_i \in [0, H+1)\ \forall i \in V$, lo que siempre se puede hacer pues pueden repetirse valores si queremos. Entonces, una interpretación geométrica directa viene al considerar que un nodo $i \in V$ pertenece al conjunto T hasta el instante $\alpha_i - 1$ y que pertenecerá a S a partir del momento α_i. Formalmente hablando:

Definición 2.2.10. Sea $R = ((V, A), (c, \tau))$ una red, H un horizonte temporal y $\alpha_i \in [0, \alpha_t)\ \forall i \in V$ con $\alpha_t = H + 1$. Entonces un *corte dinámico* $\langle S, T\rangle$ de la red R puede ser definido mediante la sucesión de cortes estáticos siguiente:

$$\langle S, T\rangle := \{\langle S_\theta, T_\theta\rangle \ : \ \theta \in [0, H]\},$$

$$\text{con } S_\theta := \{i \in V : \alpha_i \leq \theta\}, \quad T_\theta := \{i \in V : \alpha_i > \theta\}.$$

Ejemplo 2.2.11. Consideremos la red de la Figura 2.7. Para esta habíamos obtenido que su flujo máximo tenía valor 21 gracias a la construcción de un TRF y vimos que ese flujo era máximo por un corte estático en la red expandida de la Figura 2.8. Veamos qué aspecto tiene un corte dinámico mínimo en esta red.

Primero fijamos $\alpha_s = 0$ y $\alpha_t = H + 1 = 6$. Como todos los valores de τ son enteros, podemos restringir los α_i a los enteros. Si tomamos $\alpha_a = 5$ y $\alpha_b = 1$ obtenemos un corte dinámico con capacidad 21 y cuya sucesión de cortes $\langle S, T\rangle$ será:

$$\langle S_0, T_0\rangle = \langle\{s\}, \{a, b, t\}\rangle,$$

$$\langle S_\theta, T_\theta\rangle = \langle\{s, b\}, \{a, t\}\rangle, \quad \theta = 1, 2, 3, 4,$$

$$\langle S_5, T_5\rangle = \langle\{s, a, b\}, \{t\}\rangle.$$

Geométricamente lo podemos expresar como la sucesión de cortes estáticos de la Figura 2.11.

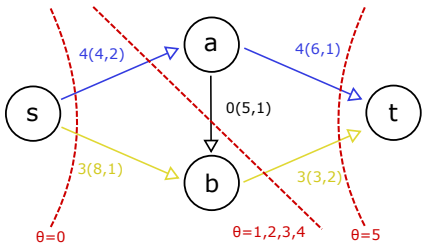

Figura 2.11: Red R con la sucesión de cortes estáticos en cada θ que definen al corte dinámico

2.3. Flujo máximo universal

Poco después de que Ford Jr y Fulkerson (1958) desarrollaron un algoritmo para calcular un flujo máximo dinámico en una red mediante TRF, Gale (1959) planteó un problema más general en referencia a los flujos máximos. Se preguntó si era posible construir un flujo dinámico que maximizase el flujo en la red para todo horizonte temporal $\theta < H$. En su artículo demostró la existencia de tal flujo, el cual denotaremos después como *flujo de llegada más temprana*.

Para tratar esta generalización del problema del flujo máximo, haremos una definición previa sobre lo que consideraremos un flujo de llegada y un flujo de salida en una red R.

Definición 2.3.1. Sea $R = ((V, A), (c, \tau))$ una red, H un horizonte temporal y H_0 el tiempo del camino más corto de s a t. Tomamos x un flujo dinámico factible en R. Llamaremos:

- *Flujo de llegada desde H_0 hasta θ.*

$$\sum_{a \in \Gamma^{-1}(t)} \sum_{z=H_0}^{\theta} x(a, z - \tau(a)) \quad \forall \theta \in \{H_0, H_0 + 1, \ldots, H\}.$$

- *Flujo de llegada desde θ hasta H.*

$$\sum_{a \in \Gamma^{-1}(t)} \sum_{z=\theta}^{H} x(a, z - \tau(a)) \quad \forall \theta \in \{H_0, H_0 + 1, \ldots, H\}.$$

- *Flujo de salida desde 0 hasta θ.*

$$\sum_{a \in \Gamma(s)} \sum_{z=0}^{\theta} x(a, z) \quad \forall \theta \in \{0, 1, \ldots, H - H_0\}.$$

- *Flujo de salida desde θ hasta $H - H_0$.*

$$\sum_{a \in \Gamma(s)} \sum_{z=\theta}^{H-H_0} x(a, z) \quad \forall \theta \in \{0, 1, \ldots, H - H_0\}.$$

Recordemos que, en general, se tiene la equivalencia entre flujos en las redes R y R_H siguiente: $x((i, j), \theta) = x_H(i_\theta, j_{\theta+\tau(i,j)}) \ \ \forall (i, j) \in A, \ i, j \in V$.

Considerando la expansión temporal hasta H de la red R y tomando nodos fuente s y destino t como los de la Figura 2.12, entonces podemos rehacer la definición anterior en términos de la red R_H, lo que nos dará una interpretación más intuitiva de lo que son estos flujos de llegada y salida.

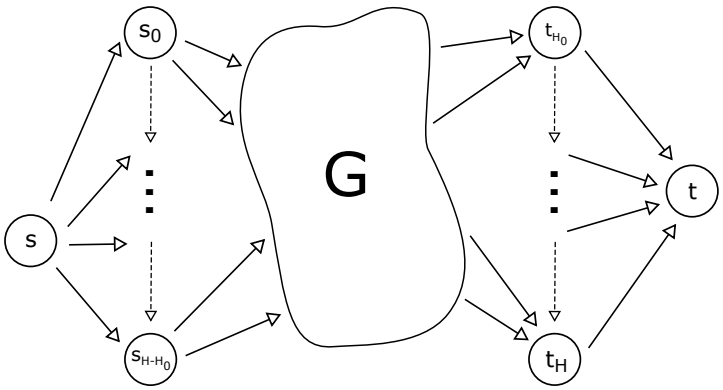

Figura 2.12: Red expandida temporalmente con arcos entre los nodos s y t y todas las copias temporales posibles

Proposición 2.3.2. *Sea $R = ((V, A), (c, \tau))$ una red, H un horizonte temporal y H_0 el tiempo del camino más corto de s a t. Tomamos x un flujo factible dinámico en R y x_H el flujo estático equivalente en la red expandida R_H. Tendremos las siguientes igualdades:*

- *Flujo de llegada desde H_0 hasta θ.*

$$\sum_{z=H_0}^{\theta} x_H(t_z, t) \quad \forall \theta \in \{H_0, H_0 + 1, \ldots, H\}.$$

- *Flujo de llegada desde θ hasta H.*

$$\sum_{z=\theta}^{H} x_H(t_z, t) \quad \forall \theta \in \{H_0, H_0 + 1, \ldots, H\}.$$

- *Flujo de salida desde 0 hasta θ.*

$$\sum_{z=0}^{\theta} x_H(s, s_z) \quad \forall \theta \in \{0, 1, \ldots, H - H_0\}.$$

- *Flujo de salida desde θ hasta $H - H_0$.*

$$\sum_{z=\theta}^{H-H_0} x_H(s, s_z) \quad \forall \theta \in \{0, 1, \ldots, H - H_0\}.$$

Demostración. Es evidente que, por la construcción de la red expandida R_H y la equivalencia que existe entre x y x_H se tiene:

$$\sum_{a \in \Gamma^{-1}(t)} x(a, z - \tau(a)) = x_H(t_z, t), \qquad \sum_{a \in \Gamma(s)} x(a, z) = x_H(s, s_z) \quad \forall z.$$

Sumando en z para el conjunto correspondiente, tenemos los cuatro casos del enunciado. □

De estas definiciones obtenemos cuatro problemas, a priori independientes, y cuyo máximo nos proporciona, en el horizonte temporal H, un flujo máximo dinámico pero con las propiedades mencionadas.

Flujo de llegada/salida más temprana/tardía		
Input:	Sea $R = ((V, A), (c, \tau))$ una red, s y t los nodos fuente y destino, H un horizonte temporal.	
Llegada más temprana:	Maximizar $v_\theta(x)$	$\forall \theta \in \{H_0, H_0 + 1, \ldots, H\}$.
Llegada más tardía:	Maximizar $v_H(x) - v_{\theta-1}(x)$	$\forall \theta \in \{H_0, H_0 + 1, \ldots, H\}$.
Salida más temprana:	Maximizar $v_{\theta+H_0}(x)$	$\forall \theta \in \{0, 1, \ldots, H - H_0\}$.
Salida más tardía:	Maximizar $v_H(x) - v_{\theta+H_0-1}(x)$	$\forall \theta \in \{0, 1, \ldots, H - H_0\}$.

Vamos a probar la existencia de todos los flujos mencionados utilizando, de manera iterativa, el Algoritmo 1 de Ford y Fulkerson del capítulo anterior para encontrar sucesivos flujos máximos estáticos que cumplan las condiciones que queremos. El método que utilizan Ford y Fulkerson para hallar un flujo máximo dinámico mediante TRF no produce en general un flujo de llegada/salida más temprana/tardía.

Proposición 2.3.3. *Sea $R = ((V, A), (c, \tau))$ una red. Siempre existe un flujo de llegada más temprana para el horizonte temporal H.*

Demostración. Tomamos H_0 el tiempo del camino más corto de s a t en R. Construimos la red expandida temporalmente para H_0 y aplicamos el Algoritmo 1 para encontrar un flujo máximo estático en R_{H_0} desde s_0 hasta t_{H_0}, llamémoslo x_{H_0}.

Construimos la red expandida temporalmente para H_0+1 y volvemos a aplicar el Algoritmo 1 para encontrar un flujo máximo estático desde s_0 hasta t_{H_0+1}. Utilizaremos el flujo factible inicial x_{H_0} que llega hasta t_{H_0} y por último el arco de espera (t_{H_0}, t_{H_0+1}). El flujo generado x_{H_0+1} se encontrará gracias a una serie de caminos de aumento (ver Definición 1.2.8) en la red residual (ver Definición 1.2.7) de la red expandida temporalmente. El flujo que pasa a través del arco de espera (t_{H_0}, t_{H_0+1}) nunca pertenece a un camino de aumento en la red residual, por tanto, el flujo x_{H_0+1} es flujo máximo para H_0 y para $H_0 + 1$.

Si repetimos este proceso (ya que el razonamiento de los caminos de aumento se mantiene en cada iteración) hasta construir la red expandida temporalmente para el horizonte H, el último flujo máximo estático que obtenemos, x_H, será equivalente a un flujo máximo dinámico en R, por la equivalencia usual $x((i, j), \theta) = x_H(i_\theta, j_{\theta+\tau(i,j)})$ $\forall (i, j) \in A$, $i, j \in V$. □

Corolario 2.3.4. *Sea $R = ((V, A), (c, \tau))$ una red. Siempre existen, para un horizonte temporal H, los flujos de la Definición 2.3.1.*

Demostración. Por la proposición anterior ya tenemos un flujo de llegada más temprana. Para ver que podemos construir un flujo de llegada más tardía tenemos que fijarnos que podemos lograrlo cambiando en la demostración anterior el orden (en este caso descendente desde H hasta H_0) por el que empezamos a buscar un flujo máximo atendiendo al detalle de que, en este caso, los arcos de espera del nodo t se toman en sentido contrario, es decir, arcos de la forma $(t_{\theta+1}, t_\theta)$.

Tomamos H_0 el tiempo del camino más corto de s a t en R. Construimos la red expandida temporalmente para H con la modificación en los arcos de espera del nodo t. Aplicamos el Algoritmo 1 para encontrar un flujo máximo estático en R_H desde s_0 hasta t_H, llamémoslo x_H.

Como en la demostración anterior, repetimos el proceso para $H-1$ usando el flujo factible x_H que llega hasta t_{H-1} a través del arco de espera que hemos invertido al principio. Aplicamos el Algoritmo 1 y encontramos un flujo x_{H-1} que respeta, por la construcción de los caminos de aumento, el flujo que llega hasta t_H.

Repitiendo el razonamiento hasta H_0 obtendremos un flujo x_{H_0} en la red expandida temporalmente R_H que, por construcción, envía el máximo flujo a cada t_θ desde H hasta H_0 en orden decreciente. El flujo dinámico equivalente en la red R es un flujo de llegada más tardía.

Para los flujos de salida tendremos que considerar la red $R' = ((V, A'), (c, \tau))$ con $A' = \{(j, i) : (i, j) \in A\}$, es la misma red que R pero con todos los arcos cambiados de sentido. Aplicando la Proposición 2.3.3 y el razonamiento de los párrafos anteriores, existen en R' flujos de llegada más temprana y tardía. Invirtiendo nuevamente el sentido de los arcos y cambiando consecuentemente los flujos, hemos construido flujos de salida más temprana y tardía en la red original R. □

La manera en la que hemos probado que existen flujos de llegada/salida más temprana/tardía tiene un inconveniente cuando la red R es grande o bien cuando el horizonte temporal H es alto. En esos casos, el coste computacional de construir la red residual de la red expandida temporalmente se hace inasumible.

Por ello se idearon algoritmos paralelos a la demostración, para la obtención del flujo de llegada más temprana, cuya eficiencia era muy superior. Por ejemplo, en Wilkinson (1971, p. 1604-1606) y Minieka (1973, p. 525-526), los autores utilizan un sistema de etiquetado sucesivo de los arcos basado en caminos más cortos. Los dos algoritmos difieren en la prueba de maximalidad del flujo generado, Wilkinson lo prueba mediante cortes mínimos en la red expandida y Minieka por medio de un razonamiento de los caminos de aumento en la red residual. Nosotros nos basamos en las ideas de Minieka para desarrollar el Algoritmo 7 equivalente al que él desarrolla por etiquetado.

Observación. La única diferencia del algoritmo expuesto frente al que podemos encontrar en Minieka (1973, p. 525-526) es que el autor etiqueta los arcos según los caminos de aumento y el sentido de los arcos, mientras que nosotros construimos el flujo dinámico $x(a, z)$ de manera explícita. Los arcos etiquetados serían aquellos a los que se les suma Δ_{P_k} y los arcos a los que se les quita una etiqueta serían aquellos a los que se resta Δ_{P_k}.

Algoritmo 7 Flujo de llegada más temprana $\mathcal{O}(n^2m + nH \max_a\{c(a)\})$

Input: Red $R = ((V, A), (c, \tau))$ y horizonte temporal H
Pasos:

1. Aplicamos el Algoritmo 3, junto con su observación, para encontrar un flujo máximo a coste mínimo x_k en R para algún $k \geq 0$.
2. Durante la ejecución del algoritmo se ha construido una sucesión de caminos de aumento a coste mínimo $\{P_k\}_k$ en cada una de las redes residuales $R_{x_k} = ((V, A_{x_k}), (c_{x_k}, \tau_{x_k}))$, con $\Delta_{P_k} = \min\{c_k(a) : a \in P_k\}$ el flujo asociado y $\tau(P_k) = \sum_{a\in P_k} \tau_{x_k}(a)$ el tiempo que tarda el flujo en llegar a t.
3. Fijamos $\theta = \tau(P_0) = H_0$, el tiempo mínimo en el que se puede llegar de s hasta t. Tomamos el flujo dinámico factible $x(a, z)$ nulo.
4. Para cada k tal que $\tau(P_k) \leq \theta$ se actualiza:

$$x(a,z) = \begin{cases} x(a,z) + \Delta_{P_k} & \text{si } a^+ \in P_k,\ \forall z \in [\theta - \tau(P_k), \theta) \\ x(a,z) - \Delta_{P_k} & \text{si } a^- \in P_k,\ \forall z \in [\theta - \tau(P_k), \theta) \\ x(a,z) & \text{en otro caso.} \end{cases}$$

5. Hacemos $\theta = \theta + 1$. Si $\theta \leq H$ volver al paso 4.
6. El flujo dinámico x es un flujo de llegada más temprana en R. FIN.

Observación. Para definir para cada uno de los caminos de aumento $\{P_k\}_k$ su coste $\tau(P_k)$ y su flujo Δ_{P_k} es necesario haber construido la red residual hasta $k-1$ en el orden que indican estos caminos de aumento por coste mínimo. Si no, no es posible asegurar que el flujo que se obtiene sea a coste mínimo, aunque sí máximo por saturación de la red residual.

Ejemplo 2.3.5. Tomemos la red R de la Figura 2.13 siguiente y fijemos $H = 6$. Si aplicamos el Algoritmo 3 de Busacker-Gowen para el flujo a coste mínimo encontramos los siguientes dos caminos de aumento de flujo (en el orden en el que los presentamos):

$$P_0 = \{(s,a),(a,b),(b,t)\}, \qquad \Delta_{P_0} = 2, \qquad \tau(P_0) = 1+1+1 = 3,$$

$$P_1 = \{(s,b),(b,a),(a,t)\}, \qquad \Delta_{P_1} = 2, \qquad \tau(P_1) = 3-1+3 = 5.$$

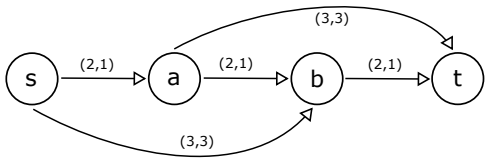

Figura 2.13: Red R con duplas en los arcos $(c(a), \tau(a))$

Notemos que el camino de aumento P_1 utiliza el arco (a, b) en sentido opuesto al original. Estos son los arcos que en el Algoritmo 7 hemos denotado por a^- en el camino de aumento. Aplicando los pasos 3, 4 y 5 del algoritmo empezando con $\theta = H_0 = 3$ tenemos:

$\theta = 3$
$\tau(P_0) \leq \theta$. Actualizamos flujo x con P_0.
$\theta = 4$
$\tau(P_0) \leq \theta$. Actualizamos flujo x con P_0.
$\theta = 5$
$\tau(P_0) \leq \theta$. Actualizamos flujo x con P_0.
$\tau(P_1) \leq \theta$. Actualizamos flujo x con P_1.
$\theta = 6$
$\tau(P_0) \leq \theta$. Actualizamos flujo x con P_0.
$\tau(P_1) \leq \theta$. Actualizamos flujo x con P_1.

En la Figura 2.15 podemos apreciar cómo se actualiza el flujo x por el algoritmo paso a paso. En azul marcamos los arcos que se han actualizado en ese paso. En amarillo los no nulos que ya teníamos en pasos anteriores. Y en rojo aquellos arcos que han sido usados en sentido contrario al original.

También podemos verlo en forma de flujo dinámico en la Figura 2.14. En esta tenemos el mismo desarrollo del flujo x pero en la red original R a lo largo del tiempo en lugar de en la red expandida temporalmente.

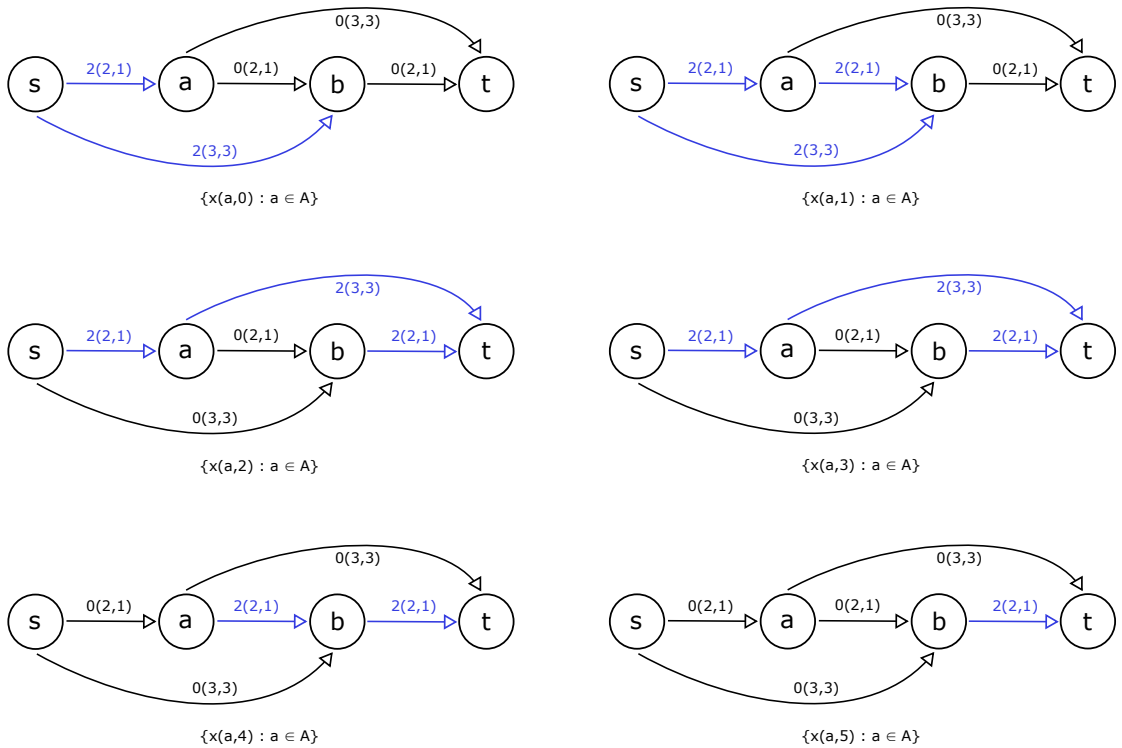

Figura 2.14: Desarrollo del flujo dinámico construido en cada instante temporal por la red R

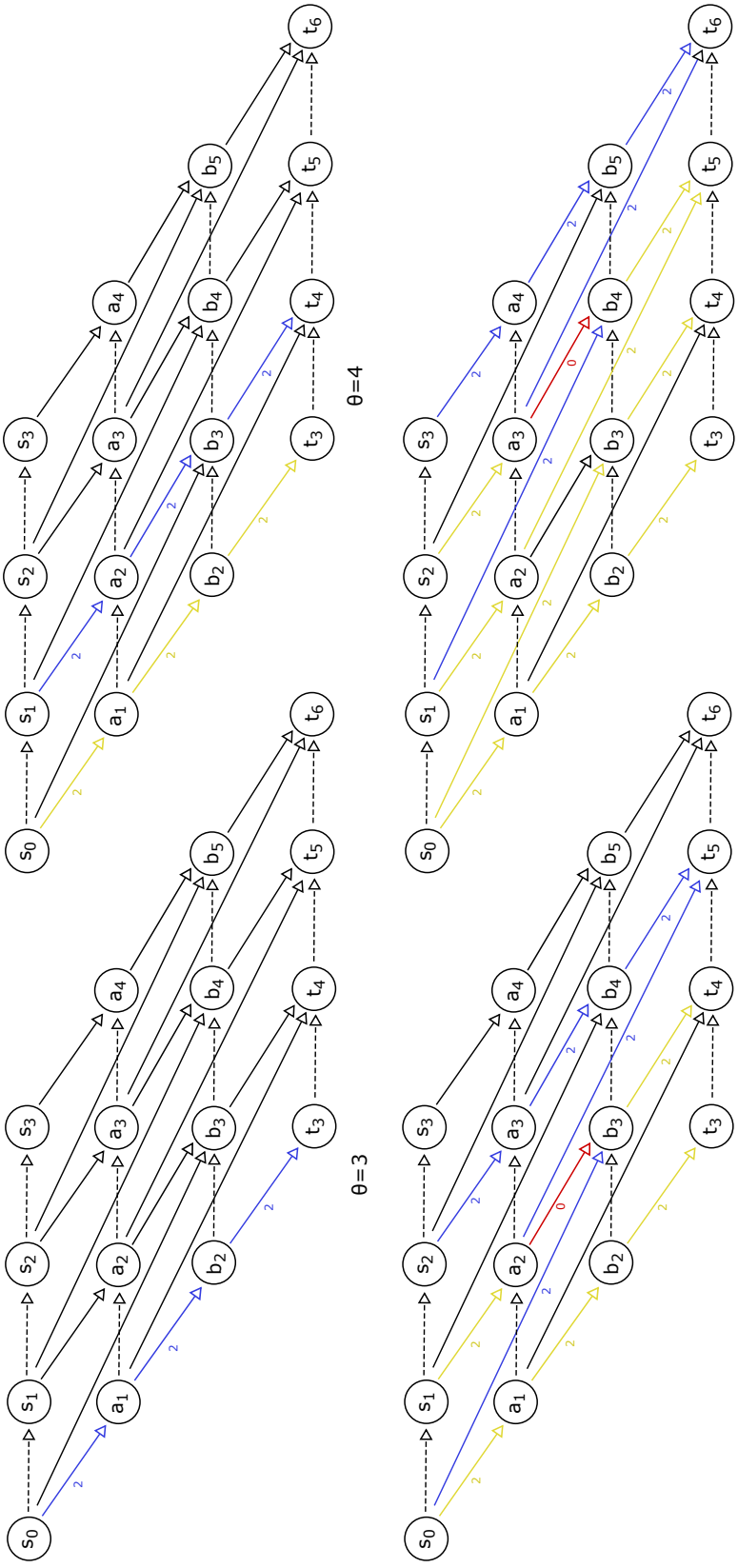

Figura 2.15: Visualización de cómo se actualiza el flujo x en la red R en cada iteración θ

Proposición 2.3.6. *El flujo resultante del Algoritmo 7 es un flujo de llegada más temprana y a su vez un flujo de salida más tardía.*

Demostración. Por construcción, para cada tiempo θ el flujo x del Algoritmo 7 es un flujo máximo por saturación de la red residual hasta ese horizonte temporal y es a coste mínimo pues los caminos de aumento que se usan son a coste mínimo. Por tanto, el flujo es máximo a coste mínimo para el horizonte θ, es decir, un flujo máximo dinámico por la observación que hicimos en el Teorema 2.2.7 sobre el problema de minimizar costes temporales. Como esto es para cada $\theta \leq H$, el flujo es de llegada más temprana.

Para ver que es de salida más tardía basta con ver que para $\theta = H$ existe el camino de aumento de flujo P_0, con $\tau(P_0) = H_0$, que envía flujo desde s en tiempo $H - H_0$ hasta t en tiempo H y, al ser el camino más corto (mínimo en coste temporal), envía flujo desde el momento más tardío posible en s. Es decir, es un flujo de salida más tardía en la red. □

Como hemos visto, al intentar construir un flujo de llegada más temprana hemos construido uno que tiene además la propiedad de salida más tardía. Es lícito preguntarse si es posible encontrar otras combinaciones de entre los cuatro problemas descritos en esta sección.

Teorema 2.3.7. *Sean x e x' dos flujos máximos dinámicos en R para un horizonte temporal H. Entonces existe un flujo máximo dinámico x'' en R para el horizonte H tal que:*

$$x''(a,z) = x(a,z) \quad \forall a \in \Gamma(s),\ z \in [0,H],$$

$$x''(a,z) = x'(a,z) \quad \forall a \in \Gamma^{-1}(t),\ z \in [0,H].$$

Es decir, tiene el mismo calendario de salida y de llegada que los dos flujos x y x'.

Demostración. Tomamos cualquier corte mínimo $\langle S, T\rangle$ en R_H. Puesto que x y x' son flujos máximos, ambos saturan el corte totalmente. Construimos el flujo x'' híbrido a partir de este corte:

$$x''((i,j),z) = \begin{cases} x((i,j),z) & \text{si } i \in S,\ \forall z \in [0,H] \\ x'((i,j),z) & \text{si } j \in T,\ \forall z \in [0,H]. \end{cases}$$

Al coincidir en el corte los dos flujos, x'' es un flujo factible bien definido y al ser x y x' máximos, x'' también lo es. □

Corolario 2.3.8. *Sea R una red, existen flujos que cumplen las siguientes condiciones:*

- *Flujo de llegada más temprana y salida más temprana.*
- *Flujo de llegada más temprana y salida más tardía.*
- *Flujo de llegada más tardía y salida más temprana.*
- *Flujo de llegada más tardía y salida más tardía.*

Demostración. Por el Corolario 2.3.4 existen los cuatro flujos por separado y utilizando el Teorema 2.3.7 podemos construir las combinaciones. □

Definición 2.3.9. Sea R una red, diremos que un flujo x es un **flujo máximo universal** si cumple cualquiera de los puntos del corolario anterior.

El Algoritmo 7 genera un flujo máximo universal puesto que, por la Proposición 2.3.6, sabemos que es un flujo de llegada más temprana y salida más tardía simultáneamente.

El algoritmo presentado para encontrar un flujo de llegada más temprana tiene un coste computacional de $\mathcal{O}(n^2m + nH\max_a\{c(a)\})$, donde $n\max_a\{c(a)\}$ es una cota al número máximo de caminos de aumento en la red R. Debido al término del máximo de las capacidades, el algoritmo alcanza el flujo en tiempo *pseudopolinomial* (ver Anexo A.1). Posteriormente se han presentado mejoras más eficientes del algoritmo, destacaremos la de Hoppe y Tardos (1994, p. 437) la cual consigue encontrar una $(1+\epsilon)$-aproximación del flujo de llegada más temprana en tiempo polinomial. Y más recientemente Baumann y Skutella (2009) construyeron un *algoritmo exacto* en tiempo polinomial para el problema del flujo de llegada más temprana en el caso de tener varias fuentes y varios destinos, el cual sería una generalización del presentado aquí.

El problema del flujo de llegada más temprana resulta ser más adecuado en la vida real, en particular en una evacuación, que el del flujo máximo dinámico. Al estar maximizando el flujo para cada paso del tiempo estamos logrando salvar al mayor número de personas posibles en cada decisión que se tome y esto es una ventaja contra posibles imprevistos, pues, si nuestro horizonte temporal es H pero ocurre una situación de emergencia imprevista en tiempo $\theta < H$, entonces desearemos haber salvado al mayor número de personas hasta ese momento.

Aunque por la Proposición 2.3.6 sepamos que el algoritmo de llegada más temprana presentado en esta sección es también de salida más tardía, podemos construir un algoritmo de llegada más temprana y salida más temprana usando el Corolario 2.3.8. Dicho flujo sería incluso más atractivo en ciertas ocasiones para evacuar si, por ejemplo, en el nodo fuente hubiera algún tipo de peligro como fuego, peligro de derrumbamiento, etc.

2.4. Flujo más rápido

Es muy habitual que en problemas como los de evacuación el flujo máximo no nos sirva para llevar a cabo la evacuación en sí, sino su utilidad resida en la planificación de la evacuación (límites de un edificio, salidas que deben hacerse, etc.). En una evacuación nos puede interesar hacer salir a los individuos lo antes posible. Digamos que en un edificio sabemos que hay, al menos, K personas. Entonces, al no saber cuántas hay en realidad nos puede interesar intentar evacuar a al menos K en un tiempo lo más pequeño posible. Este sería el marco del problema del flujo más rápido en una red.

Una versión previa del problema del flujo más rápido es la del *camino más rápido*, en la que se busca un flujo fijo v que recorra una única ruta de una red dada. Esta especificación del problema se puede dar en evacuación cuando, por limitaciones del entorno, solo se pueda escoger un camino. Por ejemplo, en el caso de un guía que intente salvar a las personas de un lugar en el que solo él conoce los caminos de la red. En ese caso el flujo debe ser concentrado por una única ruta para que el resto de personas no se pierdan durante el proceso de evacuación. Chen y Chin (1990) introducen un algoritmo que resuelve esta variante en tiempo $\mathcal{O}(m^2 + nm\log(m))$.

No hay que confundir el *flujo más rápido* con el *flujo más anticipado* (o temprano). El flujo más rápido empieza a contar el tiempo tras la primera acción que se tome, mientras que el flujo más anticipado es el que cumple la tarea en el mínimo tiempo, contando desde el momento en el que se plantea.

Ejemplo 2.4.1. Imaginemos que hay dos líneas de autobús que conectan una zona inicial, nodo s, con un destino t. Una línea tiene una frecuencia de paso alta, pero toma una ruta más larga. La otra es una línea "express" y tarda mucho menos tiempo en llegar al destino, pero pasa pocas veces al día. Entonces tenemos dos opciones, bien tomar el primer autobús que pase, que probablemente será el que toma la ruta más larga y con el que probablemente llegaremos antes, o esperar hasta que llegue el autobús "express" con el que tardaremos menos en el trayecto. El primero sería la solución más anticipada (o temprana) y el segundo autobús sería la solución más rápida.

Si hubiera algún tipo de obstáculo temporal en la red, entonces podría ser útil esperar y empezar la tarea que se quiera desarrollar más tarde, cuando el obstáculo ya no se encuentre en la red (flujo más rápido). Sin embargo, si la red no cambia a lo largo del tiempo, entonces esperar a tomar la primera acción no tiene sentido y la solución más rápida y la más anticipada son la misma. Esta será la consideración que haremos en este trabajo, aunque en evacuación siempre será recomendable trabajar con modelos que traten situaciones imprevistas y dinámicas a lo largo del tiempo.

Una subclase de soluciones del problema del flujo más rápido son aquellas con un *mínimo delay*, esto es, encontrar un flujo que minimice el tiempo que se queda en espera en los nodos. Es interesante plantearlo para aquellas situaciones en las que los nodos no son 100% seguros y se desearía que el flujo quedase parado en ellos lo mínimo posible.

Por tanto, el problema del flujo más rápido que vamos a estudiar nos pide encontrar un flujo dinámico que sea mayor que una cota dada en el menor tiempo posible en una red que no cambia con el tiempo. En algún sentido se puede ver como el inverso del problema del flujo máximo dinámico ya que, en ese, fijábamos el tiempo y maximizábamos el flujo.

FLUJO MÁS RÁPIDO

Input: Sea $R = ((V, A), (c, \tau))$ una red, s y t los nodos fuente y destino, K una constante.

Pregunta: Minimizar H sujeto a $v_H(x) \geq K$.

Llamaremos $H_K := \min\{H : v_H(x) \geq K,\ x \text{ es factible}\}$ al tiempo de la solución óptima del flujo más rápido.

Podemos observar que, si resolvemos el problema del flujo más rápido, este nos proporciona una cota inferior H_K para el horizonte temporal H con un valor de flujo de al menos K. Esta cota temporal se puede usar en el problema del flujo máximo dinámico para obtener el flujo máximo, de al menos K, de la manera más rápida. Es decir, $\max_x v_{H_K}(x) \geq K$, y la solución seguiría siendo un flujo más rápido con óptimo H_K.

Una manera para encontrar el flujo más rápido sería ir calculando el flujo máximo dinámico para diferentes horizontes temporales θ hasta que, eventualmente, uno de ellos toma un valor $v_{\theta'}(x) \geq K$. Ese θ' sería una cota superior si nos encontrásemos en tiempo continuo pero, al haber discretizado, veremos que $H_K = \theta'$.

Lema 2.4.2. *Sea $R = ((V, A), (c, \tau))$ una red, x' un flujo factible dinámico con valor K en tiempo H. Si*

$$\max_x v_{H-1}(x) < K,$$

entonces x' es un flujo más rápido con valor K y el tiempo mínimo será $H_K = H$.

Demostración. Supongamos que x' no fuese el flujo más rápido, entonces $H_K < H$. Pero como estamos en tiempo discreto y el flujo máximo dinámico en tiempo $H - 1$ tiene valor menor estricto que K se sigue que $H - 1 < H_K$. Lo cual es una contradicción, es decir, x' debía ser flujo más rápido. □

Observación. Fijémonos que $\max_x v_{H_K}(x) \geq K$ siempre y la desigualdad será estricta cuando tengamos algún arco sin saturar, es decir, cuando un corte de capacidad mínima no coincida con el valor del flujo.

Definamos como $\mathcal{F}^H$ la clase de flujos estáticos que inducen un TRF de máximo valor para un horizonte temporal H. Sabemos que siempre existe al menos uno en dicha clase, $\mathcal{F}^H \neq \emptyset$, gracias a la descomposición por caminos de Ford Jr y Fulkerson (1958) y el Teorema 2.2.7.

Ejemplo 2.4.3. Consideremos la red de la Figura 2.16. En ella construimos dos flujos con diferentes colores, el flujo x será el amarillo (red de abajo a la derecha) y el flujo y será el azul (red de arriba a la derecha).

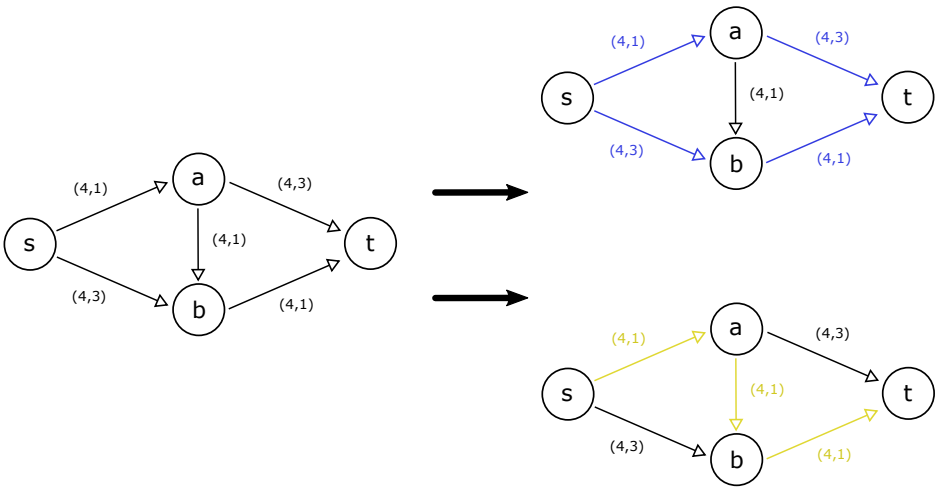

Figura 2.16: Red R con capacidades y costes temporales $(c(a), \tau(a))$

Se puede ver fácilmente que, si construimos el TRF de cada uno de ellos $\bar{x}$ e $\bar{y}$ respectivamente para el horizonte temporal $H = 4$, se obtienen los siguientes valores:

$$v(x) = 4, \quad v_4(\bar{x}) = (4+1) \cdot 4 - 12 = 8,$$

$$v(y) = 8, \quad v_4(\bar{y}) = (4+1) \cdot 8 - 32 = 8.$$

Por tanto, x e y generan cada uno un TRF de valor máximo (es fácil comprobar que es máximo con un corte mínimo) para tiempo $H = 4$. Luego $x, y \in \mathcal{F}^4$. Sin embargo, el flujo máximo estático en la red es el flujo y. Si repetimos el cálculo para $H = 3$ y para $H = 5$ es fácil ver que:

$$v_3(\bar{x}) = 4, \quad v_3(\bar{y}) = 0,$$

$$v_5(\bar{x}) = 12, \quad v_5(\bar{y}) = 16.$$

Luego, $x \in \mathcal{F}^3$ pero $y \in \mathcal{F}^5$. Además, notemos que el mínimo tiempo que utiliza el flujo y para llegar hasta el nodo t es 4, lo que tiene sentido con el valor del flujo $\bar{y}$ en tiempo 3, pues no llega flujo alguno en ese horizonte temporal.

Del Ejemplo 2.4.3 se desprende que el conjunto de los flujos máximos estáticos a coste mínimo no tiene que ser igual que $\mathcal{F}^H$ para algún H, pero siempre está contenido en él.

Presentamos a continuación unas propiedades de la función de flujo máximo respecto del horizonte temporal, que nos permitirán caracterizar su forma y crear un algoritmo para el flujo más rápido. Recordemos primero que $\mathcal{P}$ es el conjunto de todos los caminos en la red R desde el nodo fuente s al destino t y $H_0 := \min\{\tau(P) : P \in \mathcal{P}\}$ es el tiempo del camino más corto.

Proposición 2.4.4. *Sea $R = ((V, A), (c, \tau))$ una red y H un horizonte temporal. Entonces se cumplen las siguientes propiedades:*

1. *La función $f(H) := \max_x v_H(x)$ es monótona creciente y para $H \geq H_0$ es estrictamente creciente.*
2. *La función $\Delta(H) := f(H) - f(H-1)$ es monótona creciente para $H > 0$, es decir,*

$$\Delta(H) \leq \Delta(H+1) \quad \forall H > 0.$$

3. *$\Delta(H)$ está acotado superiormente por $\bar{v} = \max_x v(x)$ el valor de un flujo máximo estático en R.*

Para probar la proposición anterior vamos a hacer uso de un par de lemas que condensarán el grueso de todo el razonamiento.

Lema 2.4.5. *Sea $R = ((V, A), (c, \tau))$ una red y $f(H) := \max_x v_H(x)$. Se tiene que:*

$$f(H-1) < f(H) \quad \forall H \geq H_0.$$

Demostración. Si $H = H_0$ la desigualdad se cumple de manera obvia por la definición de H_0, ya que $v_{H_0-1}(x) = 0$ al no poder existir el flujo x en tiempo $H_0 - 1$. Tomemos $H > H_0$ y sea $y \in \mathcal{F}^{H-1}$ con valor $v(y) = \sum_{P\in\mathcal{P}} v(y_P)$ por el Lema 2.2.5 e $\bar{y}$ su TRF. Por la Definición 2.2.3 se tiene que:

$$v_{H-1}(\bar{y}) := \sum_{P\in\mathcal{P}} (H - 1 - \tau(P) + 1)v(y_P) = v_H(\bar{y}) - v(y).$$

Por tanto, como y existe y es un flujo no trivial, se tiene que:

$$\max_x v_H(x) \geq v_H(\bar{y}) = v_{H-1}(\bar{y}) + v(y) > v_{H-1}(\bar{y}) = \max_x v_{H-1}(x).$$

□

Lema 2.4.6. *Sea $R = ((V, A), (c, \tau))$ una red, H un horizonte temporal y $f(H) := \max_x v_H(x)$. Tomamos $y \in \mathcal{F}^H$ y $z \in \mathcal{F}^{H+1}$. Se cumplen:*

1. $v(y) \leq v(z) \quad \forall H \geq 0.$
2. $v(y) + f(H) \leq f(H + 1) \leq v(z) + f(H) \quad \forall H \geq 0.$

Demostración. Puesto que y, z son flujos factibles estáticos en R y cada uno induce un TRF máximo hasta los horizonte H y $H + 1$ respectivamente, se tiene:

$$v_{H+1}(\bar{z}) \geq v_{H+1}(\bar{y}),$$

$$v_H(\bar{z}) \leq v_H(\bar{y}).$$

Restamos la segunda desigualdad a la primera y, usando la expresión del valor de un TRF dada en el Teorema 2.2.7, se obtiene que:

$$v_{H+1}(\bar{y}) - v_H(\bar{y}) = v(y) \leq v(z) = v_{H+1}(\bar{z}) - v_H(\bar{z}),$$

lo que prueba 1. Por la Definición 2.2.3 y el Lema 2.2.5 aplicados a y se tiene que:

$$v_{H+1}(\bar{y}) := \sum_{P\in\mathcal{P}} (H + 1 - \tau(P) + 1)v(y_P) = v_H(\bar{y}) + v(y).$$

Entonces como $y \in \mathcal{F}^H$:

$$v(y) + \max_x v_H(x) = v(y) + v_H(\bar{y}) = v_{H+1}(\bar{y}) \leq \max_x v_{H+1}(x).$$

Otra vez, por la Definición 2.2.3 y el Lema 2.2.5 aplicados a z se tiene que:

$$v_{H+1}(\bar{z}) := \sum_{P\in\mathcal{P}} (H + 1 - \tau(P) + 1)v(z_p) = v_H(\bar{z}) + v(z).$$

Entonces como $z \in \mathcal{F}^{H+1}$:

$$\max_x v_{H+1}(x) = v_{H+1}(\bar{z}) = v(z) + v_H(\bar{z}) \leq v(z) + \max_x v_H(x),$$

lo que demuestra 2. □

Demostración (Proposición 2.4.4). La prueba de 1 es directa por el Lema 2.4.5. Para la prueba de 2 solo debemos fijarnos que, por el Lema 2.4.6 aplicado a $x \in \mathcal{F}^{H-1}, y \in \mathcal{F}^{H}, z \in \mathcal{F}^{H+1}$, tenemos

$$v(x) \leq \Delta(H) \leq v(y) \leq \Delta(H+1) \leq v(z).$$

El punto 3 es, con la expresión anterior, obvio:

$$0 \leq v(x) \leq \Delta(H) \leq v(y) \leq \bar{v}.$$

□

Estas propiedades de la función del valor de un flujo máximo son utilizadas por Burkard et al. (1993, p. 39-44). En su trabajo se proporcionan una serie de algoritmos para resolver el problema del flujo más rápido en tiempo polinomial. Por el Teorema 2.4.10, que veremos después, y el Algoritmo 7, ya podríamos haber construido un flujo más rápido en una red R en tiempo pseudopolinomial mediante un flujo de llegada más temprana.

Presentaremos aquí dos de los algoritmos que se presentan en Burkard et al. (1993), uno basado en una búsqueda binaria con una mejora por la forma específica de la función f de la Proposición 2.4.4, y otro basado en el método de Newton adaptado a la función. La idea detrás de ambos es buscar H el mínimo valor para el cual $v_H(x) \geq K$ y, utilizando el Lema 2.4.2, llegar a la conclusión de que ese H es el óptimo del flujo más rápido.

Como tenemos que buscar valores en tiempo discreto y la función $f(H) := \max_x v_H(x)$ es monótona creciente respecto de H (ver Proposición 2.4.4), un método de búsqueda binaria parece una buena primera opción. Se empieza con un intervalo $[H_l, H_u]$ tal que $f(H_l) < K$ y $f(H_u) > K$. Entonces se toma el punto medio del intervalo $\lfloor (H_u + H_l)/2 \rfloor$ y se calcula el flujo máximo para ese horizonte temporal. Si el valor está por encima de K, se selecciona la primera mitad del intervalo, y si está por debajo, la segunda mitad.

Este método se puede acelerar usando que, por la propiedad 2 de la Proposición 2.4.4, la función f es convexa y tiene una forma que podemos aprovechar para mejorar los intervalos en los que buscamos el flujo máximo.

Como f solo está definida para los valores enteros, si la extendiésemos a toda la recta real, su gráfica sería la de una función lineal a trozos creciente y convexa (ver Figura 2.17). Introduciremos la siguiente notación que nos guiará en la comprensión de la mejora de los intervalos que queremos construir. Sea L_K la línea paralela al eje temporal de H a una altura K, siendo K la cota mínima del valor del flujo más rápido que queremos determinar. Supondremos que $H_K \in [H_l, H_u]$ (valor óptimo del problema del flujo más rápido) y, por ello, tiene sentido definir la recta que une los puntos del plano $(H_l, f(H_l))$ y $(H_u, f(H_u))$ y que cortará a la línea L_K por hipótesis. La coordenada del eje temporal del punto de intersección (intersección de la línea azul y roja en la Figura 2.17) lo denotaremos por $\gamma(H_l, H_u)$.

Tomemos la recta $L_{sg,H}$ que pasa por el punto $(H, f(H))$ y es tangente a la gráfica de f en ese punto. Al ser f lineal a trozos esta recta no tiene que ser única y vendrá representada por un subgradiente de la función en ese punto.

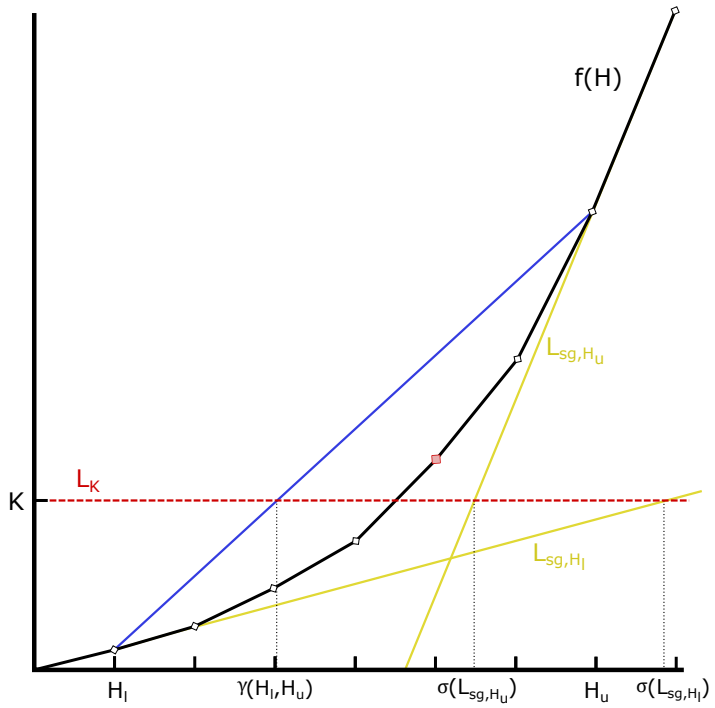

Figura 2.17: Esquema de la mejora en la búsqueda binaria. Punto óptimo marcado en rojo

Lema 2.4.7.

1. *Si $y \in \mathcal{F}^H$, entonces $v(y)$ es subgradiente de la función f en el punto H.*
2. *Si $y \in \arg\min\{v(x) : x \in \mathcal{F}^H\}$, entonces $\Delta(H) = v(y)$.*

Demostración. Recordemos que si $f : I \longrightarrow \mathbb{R}$ es una función convexa en un abierto $I \subset \mathbb{R}$, entonces $c \in \mathbb{R}$ es un subgradiente de f en el punto $x_0 \in I$ si

$$f(x) - f(x_0) \geq c(x - x_0) \quad \forall x \in I.$$

1. Tomando $y \in \mathcal{F}^H$ y usando el segundo punto del Lema 2.4.6, tenemos que

$$\Delta(H+1) = f(H+1) - f(H) \geq v(y),$$

donde $v(y)$ es el valor del flujo estático y. Por la definición de subgradiente, $v(y)$ lo es de f en el punto H.

2. Por la Proposición 2.4.4 se tiene que $\Delta(H+1) \geq \Delta(H)$, por tanto, $\Delta(H)$ es también un subgradiente de f en H. Tomemos $y \in \arg\min\{v(x) : x \in \mathcal{F}^H\}$. Al ser $v(y)$ el menor de los subgradientes se tiene que $\Delta(H) \geq v(y)$. Veamos que la desigualdad estricta no se puede dar. Supongamos que $\exists z \in \mathcal{F}^H$ tal que $v(z) < \Delta(H)$. Entonces:

$$f(H) - v_{H-1}(\bar{z}) = v(z) < \Delta(H) = f(H) - f(H-1),$$

de lo que se tiene que $f(H-1) < v_{H-1}(\bar{z})$, lo cual es una contradicción pues $f(H-1) := \max_x v_{H-1}(x)$. □

Consideremos, finalmente, los puntos de intersección de las rectas L_{sg,H_l} y L_{sg,H_u} con la recta L_K (intersección de las líneas amarillas y roja en la Figura 2.17). Denotaremos la coordenada del eje temporal de los puntos de intersección como $\sigma(L_{sg,H_l})$ y $\sigma(L_{sg,H_u})$ respectivamente. De la convexidad de f se pueden obtener las siguientes cotas con la notación utilizada:

$$\lceil \gamma(H_l, H_u) \rceil \leq H_K \leq \lceil \min\{\sigma(L_{sg,H_l}), \sigma(L_{sg,H_u})\} \rceil.$$

La parte izquierda se deduce, con la recta definida antes desde $(H_l, f(H_l))$ hasta $(H_u, f(H_u))$, usando un razonamiento como el de la regula falsi. Mientras que la parte derecha, al estar usando el subgradiente, puede obtenerse por medio de un razonamiento similar al del método de Newton.

Algoritmo 8 Flujo más rápido (Binario+) $\mathcal{O}(n^2 m \min\{\log(K), n \max_a\{c(a)\}\})$

Input: Red $R = ((V, A), (c, \tau))$ y cota inferior del flujo K
Pasos:

1. Se construye un *buen* intervalo $[H_l, H_u]$ que contenga a H_K.
2. Se calculan sendos flujos a coste mínimo con el Algoritmo 3 en las redes modificadas añadiendo el arco $\{(t, s)\}$ con $c(t, s) = \infty$, $\tau(t, s) = -(H_l + 1)$ para una y $\tau(t, s) = -(H_u + 1)$ para la otra. Los flujos x_l, x_u obtenidos cumplen que $x_l \in \mathcal{F}^{H_l}$, $x_u \in \mathcal{F}^{H_u}$.
3. Se calculan los puntos $(H_l, f(H_l))$, $(H_u, f(H_u))$ y se computan el resto de elementos de la mejora binaria:

$$\gamma(H_l, H_u) := \frac{(H_u - H_l)(K - f(H_l))}{f(H_u) - f(H_l)} + H_l,$$

$$\sigma(L_{sg,H_l}) := \frac{K - f(H_l)}{v(x_l)} + H_l, \quad \sigma(L_{sg,H_u}) := \frac{K - f(H_u)}{v(x_u)} + H_u.$$

4. Se actualizan las cotas del intervalo y se calcula el punto medio:

$$H_l := \lceil \gamma(H_l, H_u) \rceil, \quad H_u := \lceil \min\{\sigma(L_{sg,H_l}), \sigma(L_{sg,H_u})\} \rceil, \quad H_m := \left\lfloor \frac{H_l + H_u}{2} \right\rfloor.$$

5. Se aplica el Algoritmo 3 para calcular un flujo a coste mínimo con $\tau(t, s) = -(H_m + 1)$. Se calcula $f(H_m)$ directamente.
6. Si $f(H_m) > K$, entonces $H_u = H_m$. Si $f(H_m) < K$, entonces $H_l = H_m$. En caso de que $f(H_m) = K$, se tiene que $H_K := H_m$ y FIN.
7. Si $H_u - H_l \leq 1$, entonces $H_K := H_u$ y FIN. En caso contrario, volver al paso 2.

En el primer paso del Algoritmo 8 se dice que hay que calcular unos buenos valores para $[H_l, H_u]$. Vamos a dar un par de cotas que siempre contienen el tiempo objetivo H_K entre ellas y cuyo cálculo no domina el resto del algoritmo.

$$H_l := H_0 + \left\lceil \frac{K - f(H_0)}{\overline{v}} \right\rceil, \quad H_u := H_l + \left\lceil \frac{K - f(H_l)}{v(x^l)} \right\rceil.$$

- Supondremos para las cotas que f es una función continua y definida en todos los reales, de modo que se alcanza $K = f(H_K)$. Al tomar enteros, esta asunción no tiene efecto negativo en nuestras cotas.
- H_0 es el tiempo del camino más corto de s a t en la red.
- $\bar{v} = \max_x v(x)$ es el valor de un flujo máximo estático.
- $v(x_l)$ es el valor de un flujo estático $x_l \in \mathcal{F}^{H_l}$.
- $f(*)$ es el valor del flujo máximo dinámico en tiempo $H = *$.

Lema 2.4.8. *Con H_l y H_u así definidos, siempre se tiene $H_K \in [H_l, H_u]$.*

Demostración. Por la tercera propiedad de la Proposición 2.4.4 se tiene que:

$$\Delta(H) \leq \bar{v} \;\Rightarrow\; \frac{f(H) - f(H-1)}{\bar{v}} \leq 1.$$

Como esta desigualdad se cumple para todo horizonte temporal H, vamos a hacer un sumatorio desde $H_0 + 1$ hasta H_K:

$$\sum_{\theta=H_0+1}^{H_K} \frac{f(\theta) - f(\theta-1)}{\bar{v}} \leq H_K - H_0.$$

Como el término de la izquierda es una suma telescópica nos queda:

$$\frac{f(H_K) - f(H_0)}{\bar{v}} \leq H_K - H_0.$$

Suponiendo que se alcanza $f(H_K) = K$ en tiempo H_K continuo, se tiene que

$$H_0 + \left\lceil \frac{K - f(H_0)}{\bar{v}} \right\rceil \leq H_K.$$

Por otro lado, tomemos $z \in \mathcal{F}^{H_l}$. Por la segunda propiedad de la Proposición 2.4.4 se tiene:

$$\Delta(H_K) \geq \cdots \geq \Delta(H_l + 2) \geq \Delta(H_l + 1) = f(H_l + 1) - f(H_l) \geq v(z).$$

Si sumamos desde H_l hasta $H_K - 1$ se tiene:

$$\sum_{\theta=H_l}^{H_K-1} \frac{f(\theta+1) - f(\theta)}{v(z)} \geq H_K - H_l.$$

Suponiendo que se alcanza $f(H_K) = K$ en tiempo H_K continuo, se tiene que

$$H_l + \left\lceil \frac{K - f(H_l)}{v(z)} \right\rceil \geq H_K.$$

□

El cálculo de estas cotas se hace por separado del bucle y necesita el cálculo de un flujo máximo estático, un flujo a coste mínimo, un camino más corto y un par de flujos dinámicos que se pueden calcular a partir de lo ya calculado, en total, el número de operaciones es del orden de $\mathcal{O}(n^2(m+n))$.

El cálculo de los flujos a coste mínimo en las redes modificadas, de la forma como dijimos en la sección de flujos máximos dinámicos, es necesario, pues nos permite calcular el valor de f en H_l y H_u. También nos sirven para computar los subgradientes de las líneas L_{sg,H_l} y L_{sg,H_u}, junto con el punto $\gamma(H_l, H_u)$ que se obtiene muy fácilmente con la ecuación punto-pendiente. Todo esto tiene coste $\mathcal{O}(2n^2m)$.

A continuación, se actualizan los puntos $[H_l, H_u]$ y se calcula $H_m = \lfloor (H_l + H_u)/2 \rfloor$ su punto medio. De este se encuentra el valor $f(H_m)$, lo cual requiere un cálculo de un flujo a coste mínimo, es decir, $\mathcal{O}(n^2m)$ operaciones.

Esto se repite hasta que $f(H_m) \approx K$, haciendo $H_K = \lceil H_m \rceil$. O bien $H_u - H_l \leq 1$, en cuyo caso, en virtud del Lema 2.4.2, se tiene que $H_K = H_u$, lo que son, a lo sumo, $\min\{\lceil \log(K) \rceil + 1, \bar{v}\}$ iteraciones, puesto que el intervalo se reduce al menos hasta la mitad en cada paso y la función f tiene como mucho $\bar{v}$ cambios de pendiente por la propiedad tres de la Proposición 2.4.4.

Ejemplo 2.4.9. Si recuperamos la red de la Figura 2.16, podemos demostrar que el flujo $y \in \mathcal{F}^H \ \forall H \geq 4$. Por tanto, tenemos caracterizada la función $f(H)$ y se puede buscar fácilmente el valor H tal que $f(H-1) < K \leq f(H)$. Si no, bastaría con aplicar el algoritmo descrito, construyendo las líneas y los subgradientes que se piden en cada caso.

El otro algoritmo (ver Algoritmo 9) que se presenta está basado en el método de Newton para resolver ecuaciones no lineales. Con la notación que hemos usado, para un punto $(H^{(k)}, f(H^{(k)}))$ se considerará una cualquiera de sus rectas tangentes $L_{sg,H^{(k)}}$, se busca el punto de corte con la recta L_K y se hace la imagen por f de la coordenada temporal $\sigma(L_{sg,H^{(k)}})$. Esto, junto con el Lema 2.4.7 que nos permite calcular un subgradiente, nos proporciona la siguiente fórmula recursiva:

$$H^{(k+1)} := \sigma(L_{sg,H^{(k)}}) = H^{(k)} + \frac{K - f(H^{(k)})}{v(x^{(k)})}, \qquad x^{(k)} \in \mathcal{F}^{H^{(k)}}.$$

Al ser f una función convexa, la convergencia se tiene garantizada en un número finito de pasos y, además, en la práctica se suele obtener la solución más rápido que con el Algoritmo 8.

El cálculo de la cota inicial H_l en el Algoritmo 9 no domina en ningún caso su velocidad. Incluso si fuera muy costoso se podría tomar $H^{(0)} = H_0$ o igual a cero. Después se calcula en cada iteración del bucle un flujo a coste mínimo con el algoritmo de Busacker-Gowen que tiene $\mathcal{O}(n^2m)$. Por la tercera propiedad de la Proposición 2.4.4 tenemos que el número máximo de pendientes que tiene la función a trozos f es $\bar{v} = \max_x v(x)$. Este valor está acotado, en el peor de los casos, por $n \max_a\{c(a)\}$, lo que haría al algoritmo pseudopolinomial, ya que sería el número máximo de caminos de aumento que, a lo sumo, se tienen que usar para alcanzar un flujo máximo estático.

Algoritmo 9 Flujo más rápido (Newton) $\mathcal{O}(n^2 m \log(nK \max_a\{c(a)\}))$

Input: Red $R = ((V, A), (c, \tau))$ y cota inferior del flujo K
Pasos:

1. Hacemos $k = 0$ y tomamos $H^{(k)} := H_l$.
2. Aplicamos el Algoritmo 3 y encontramos un flujo a coste mínimo en la red modificada $R' = ((V, A \cup \{(t, s)\}), (c, \tau))$ con $c(t, s) = \infty$ y $\tau(t, s) = -(H^{(k)} + 1)$. El flujo encontrado será $x^{(k)} \in \mathcal{F}^{H^{(k)}}$ por el Teorema 2.2.7.
3. Calculamos $f(H^{(k)})$ a partir de $x^{(k)}$. Y computamos:

$$H^{(k+1)} = H^{(k)} + \frac{K - f(H^{(k)})}{v(x^{(k)})}$$

4. Si $|H^{(k)} - H^{(k+1)}| < 1$ o $f(H^{(k+1)}) = K$. Hacemos $H_K = \lceil H^{(k)} \rceil$ y FIN. En caso contrario, $k = k + 1$ y volvemos al paso 2.

En Burkard et al. (1993, p. 43) se prueba que, dada la estructura del problema, se puede rebajar la cota hasta $\mathcal{O}(\log(nK \max_a\{c(a)\}))$. Por lo tanto, nos queda una cota global del algoritmo de $\mathcal{O}(n^2 m \log(nK \max_a\{c(a)\}))$ según lo presentamos aquí. Si en lugar de buscar una solución en tiempo discreto lo hiciésemos en tiempo continuo se puede volver a hacer por medio de una búsqueda binaria ya que Fleischer y Tardos (1998, p. 76) probaron, en el caso en el que K y $\tau(a)$ son enteros para todo $a \in A$, que el tiempo óptimo de un flujo más rápido es un número racional cuyo denominador está acotado por la capacidad de un corte mínimo en la red original por el Teorema 1.2.11 y por la expresión del valor de un flujo máximo en la clase de los TRF del Teorema 2.2.7.

Teorema de triple optimización

Hay una relación muy estrecha entre el flujo de llegada más temprana y el flujo más rápido en una red. El siguiente teorema, que se llama el *Teorema de triple optimización*, se lo debemos a Jarvis y Ratliff (1982). Probaron que, si se encuentra una solución para el problema del flujo de llegada más temprana, entonces se tiene una solución para el problema del flujo más rápido dada una cota factible K. Es más, Jarvis y Ratliff demostraron que el problema del flujo de llegada más temprana es equivalente al problema de encontrar un flujo mínimo ponderado en los arcos de llegada de la red expandida hasta un horizonte H, siempre que el flujo sea de al menos K unidades.

Esta equivalencia es bastante útil si hablamos de eficiencia computacional. Encontrar un flujo de llegada más temprana es, a priori, más costoso que encontrar un flujo más rápido tanto si hablamos del coste computacional como del esfuerzo para programar el código. Si comparamos el Algoritmo 9, basado en el método de Newton, con el Algoritmo 7, que es pseudopolinomial, las ventajas del primero quedan patentes.

Teorema 2.4.10 (Triple optimización)**.** *Sea $R = ((V, A), (c, \tau))$ una red, x' un flujo máximo dinámico con valor $v_H(x') \geq K$ para un horizonte temporal H. Consideremos:*

(a) Problema del flujo de llegada más temprana:

$$\max_x \ v_\theta(x) = \sum_{z=0}^{\theta} x_H(t_z, t) \quad \forall \theta \leq H.$$

(b) Problema del mínimo flujo ponderado de llegada:

$$\min_x \sum_{z=0}^{H} z x_H(t_z, t).$$

(c) Problema del flujo más rápido:

$$\min_x \{H \ : \ v_H(x) \geq K\}.$$

Si x' es solución de uno de los problemas (a) o (b), entonces es también solución de los otros.

Observación. Los problemas tanto de (a) como de (c) ya han sido descritos con anterioridad. El problema (b) tiene una interpretación directa como el mínimo tiempo medio que tarda el flujo en llegar hasta el destino t. Esto es obvio, pues estamos ponderando el flujo según el momento temporal en el que llega al nodo t y, dividiendo el resultado entre el valor del flujo total $v_H(x)$, se obtiene un tiempo medio de salida de la red R para el primer individuo del flujo. Esto es bastante útil al planificar una evacuación, pues de media podremos asegurar un cierto tiempo de evacuación admisible.

Para demostrar el teorema necesitamos de un lema técnico previo que nos servirá para ver la equivalencia entre los apartados (a) y (b).

Lema 2.4.11. *Consideremos los conjuntos de números reales no negativos $\{\alpha_i\}_{i=1}^n, \{\beta_i\}_{i=1}^n$ y $\{\gamma_i\}_{i=1}^n$ que satisfacen:*

1. $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$.
2. $\sum_{i=1}^k \alpha_i \leq \sum_{i=1}^k \beta_i \ \forall k = 1, \ldots, n$.
3. $\gamma_1 < \gamma_2 < \cdots < \gamma_n$.

Si para al menos un $i \in \{1, \ldots, n\}$ se tiene que $\alpha_i \neq \beta_i$, entonces:

$$\sum_{i=1}^n \gamma_i \alpha_i > \sum_{i=1}^n \gamma_i \beta_i$$

Demostración. La clave de la demostración es construir una sucesión de conjuntos $\{\{\alpha_i^j\}_{i=1}^n\}_j$ con $\alpha_i^0 = \alpha_i$ tal que $\beta_i = \alpha_i^j \ \forall i$ para algún j y que dicha sucesión cumpla:

$$\sum_{i=1}^{n} \gamma_i \alpha_i^j > \sum_{i=1}^{n} \gamma_i \alpha_i^{j+1} \quad \forall j.$$

Para ello consideremos el mínimo índice

$$q_j := \min \left\{ k : \sum_{i=1}^{k} \alpha_i^j < \sum_{i=1}^{k} \beta_i \right\}.$$

Por el hecho de ser mínimo, y por la segunda condición del enunciado, se tiene que:

$$\beta_{q_j} - \alpha_{q_j}^j > 0, \qquad \beta_i = \alpha_i^j \ \forall i < q_j.$$

Construyamos

$$\alpha_i^{j+1} = \begin{cases} \alpha_i^j & \text{si } i < q_j \\ \alpha_i^j + (\beta_{q_j} - \alpha_{q_j}^j) & \text{si } i = q_j \\ \alpha_i^j - (\beta_{q_j} - \alpha_{q_j}^j) & \text{si } i = q_j + 1 \\ \alpha_i^j & \text{si } i > q_j + 1. \end{cases}$$

Al sumar y restar la misma cantidad se tiene que el conjunto $\{\alpha_i^j\}_{i=1}^n$ sigue cumpliendo la primera condición para todo j. Además, como se suma la cantidad en exceso en el índice q_j, la segunda condición también se cumple ya que $\beta_{q_j} - \alpha_{q_j}^j > 0$. Veamos que la suma ponderada decrece estrictamente:

$$\begin{aligned} \sum_{i=1}^{n} \gamma_i \alpha_i^{j+1} &= \sum_{i=1}^{q} \gamma_i \alpha_i^j + \gamma_{q_j}(\alpha_{q_j}^j + (\beta_{q_j} - \alpha_{q_j}^j)) + \gamma_{q_j+1}(\alpha_{q_j+1}^j - (\beta_{q_j} - \alpha_{q_j}^j)) + \sum_{i=q+2}^{n} \gamma_i \alpha_i^j = \\ &= \sum_{i=1}^{n} \gamma_i \alpha_i^j + (\gamma_{q_j} - \gamma_{q_j+1})(\beta_{q_j} - \alpha_{q_j}^j) < \sum_{i=1}^{n} \gamma_i \alpha_i^j. \end{aligned}$$

Puesto que $\beta_{q_j} - \alpha_{q_j}^j > 0$ y por la tercera condición $\gamma_{q_j} - \gamma_{q_j+1} < 0$.

Si repetimos el proceso hasta que no se pueda encontrar un q_j mínimo índice que hace que la suma de los α^j sea estrictamente menor que la suma de los β, esto querrá decir, por la condición dos, que todas las sumas son iguales y, en particular, que $\alpha_i^j = \beta_i \ \forall i$. Por lo tanto, se tiene el resultado del enunciado, ya que si existe $\alpha_i \neq \beta_i$ entonces hay un mínimo índice q y podemos empezar el proceso descrito arriba. □

Demostración: (Teorema 2.4.10).

$\boxed{(a) \Longrightarrow (b)}$ Supongamos que x' es solución de (a) y denotemos por $\beta_z := x'_H(t_z, t)$, el flujo x' en la red expandida R_H en el arco (t_z, t).
Tomemos x cualquier flujo máximo dinámico de R en tiempo H y denotemos por $\alpha_z := x_H(t_z, t)$. Al ser x flujo máximo e x' flujo de llegada más temprana se tiene que:

$$\sum_{z=0}^{H} \alpha_z = \sum_{z=0}^{H} \beta_z,$$

$$\sum_{z=0}^{\theta} \alpha_z \leq \sum_{z=0}^{\theta} \beta_z \quad \forall \theta \in \{0, 1, \ldots, H\}$$

Consideremos $\gamma_z := z$. Entonces, si $\alpha_z \neq \beta_z$ para algún z, por el lema anterior se tiene que

$$\sum_{z=0}^{H} \gamma_z \alpha_z > \sum_{z=0}^{H} \gamma_z \beta_z.$$

Por lo tanto, x' es flujo mínimo ponderado ya que x es un flujo máximo arbitrario.

$\boxed{(b) \Longrightarrow (a)}$ Supongamos que x' es solución de (b). Por las Proposiciones 2.3.2 y 2.3.3 sabemos que siempre existe un flujo x de llegada más temprana con $v_H(x) = v_H(x') \geq K$ por ser x' máximo en tiempo H. Denotamos por $\beta_z := x_H(t_z, t)$, $\alpha_z := x'_H(t_z, t)$ y $\gamma_z := z$.
Si existe un z tal que $\alpha_z \neq \beta_z$, entonces por el lema anterior se tiene que:

$$\sum_{z=0}^{H} \gamma_z \alpha_z > \sum_{z=0}^{H} \gamma_z \beta_z.$$

Esto contradeciría el hecho de que x' es un flujo mínimo ponderado. Por tanto, $\alpha_z = \beta_z \; \forall z$ y se tiene que el flujo x' es de llegada más temprana.

$\boxed{(a) \Longrightarrow (c)}$ Supongamos que x' satisface (a) pero que no satisface (c). Por tanto, existe un flujo x que en tiempo $\theta \in \{0, \ldots, H\}$ es flujo más rápido con $v_\theta(x) \geq K$. Como x' es flujo de llegada más temprana, se sigue que $v_\theta(x') \geq v_\theta(x) \geq K$. Por tanto, x' era también flujo más rápido en tiempo θ. □

2.5. Flujo y localización dinámica

Igual que en el caso estático, nos podemos plantear colocar q instalaciones en una red mientras se pretende que el flujo máximo hasta un horizonte temporal H sea perturbado lo menos posible. Gracias a la interpretación de un problema dinámico que nos ofrece la red expandida temporalmente, cuando tratamos de colocar q instalaciones de manera dinámica, estamos situando cada instalación en todas las copias temporales de los arcos de la red expandida hasta el horizonte H simultáneamente (ver Figura 2.18).

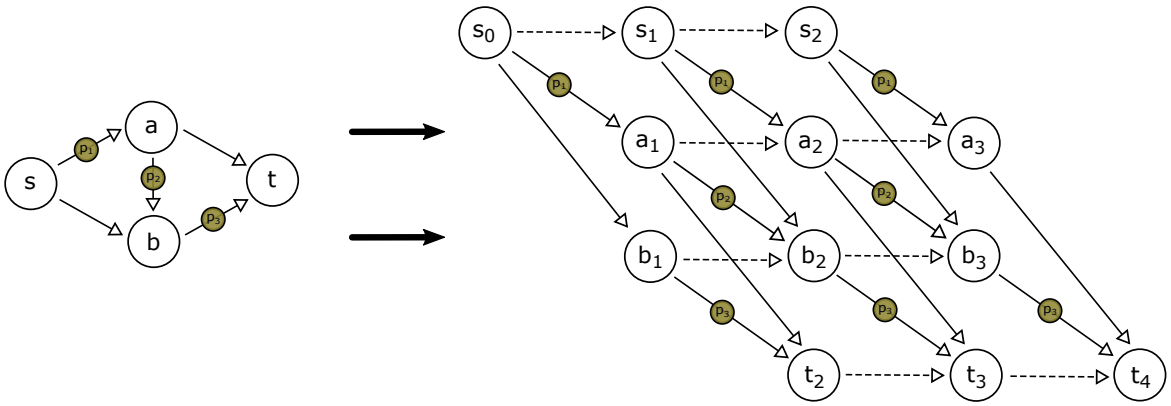

Figura 2.18: Localización de tres instalaciones $\{p_1, p_2, p_3\}$ en la red R y su colocación en la red expandida temporalmente con $H = 4$

Definición 2.5.1. Sea $R_{\mathbb{P}} = ((V, A), (c, d, m))$ una red de flujo y localización (ver Definición 1.4.1). Consideramos $\tau : A \longrightarrow \mathbb{R}$ una función de costes de tránsito por un arco y la extensión de la función de capacidades c a todos los nodos de la red $c : A \cup V \longrightarrow \mathbb{R}^+$. Se denominará *red dinámica de evacuación* o *red de flujo y localización con esperas* a

$$R_{\mathbb{P}} = ((V, A), (c, \tau, d, m)).$$

Por exceso de notación, a la red dinámica de evacuación la llamaremos simplemente red de flujo y localización y la definiremos explícitamente con la función τ de costes, sabiendo que la componente temporal se tiene en cuenta.

Al igual que el problema estático, el q-FlowLoc dinámico es $\mathcal{NP}$-completo (ver Teorema 1.4.3) y se podría plantear una formulación de programación lineal entera mixta usando la Definición 2.1.3 de flujo dinámico factible. Sin embargo, por el Teorema 2.2.7 sabemos que un flujo máximo dinámico puede ser calculado a través de un flujo estático a coste mínimo en la red original añadiendo un arco entre los nodos destino t y fuente s. Esto se puede expresar como formulación lineal como sigue:

Variables

x_a : flujo en el arco a $\quad \forall a \in A$

$y_{a,p}$: $\begin{cases} 1 \text{ si la instalación } p \text{ se coloca en el arco } a \\ 0 \text{ en otro caso.} \end{cases}$ $\quad \forall a \in A, p \in \mathbb{P}$

Constantes

d_p : tamaño de la instalación $p \in \mathbb{P}$.

c_a : capacidad del arco $a \in A$.

m_a : máximo número de instalaciones que se pueden colocar en $a \in A$.

τ_a : coste/tiempo del arco $a \in A$.

H : horizonte temporal.

Formulación

$$\text{min.} \quad \sum_{a \in A \setminus \{(t,s)\}} \tau_a x_a - (H+1) x_{(t,s)} \tag{2.4}$$

$$\text{s.a} \quad \sum_{a \in \Gamma^{-1}(i)} x_a - \sum_{a \in \Gamma(i)} x_a = 0 \qquad \forall i \in V \tag{2.4a}$$

$$0 \leq x_a \leq c_a \qquad \forall a \in A \setminus \mathbb{L} \tag{2.4b}$$

$$\sum_{p \in \mathbb{P}} y_{a,p} \leq m_a \qquad \forall a \in \mathbb{L} \tag{2.4c}$$

$$\sum_{a \in \mathbb{L}} y_{a,p} = 1 \qquad \forall p \in \mathbb{P} \tag{2.4d}$$

$$x_a + d_p y_{a,p} \leq c_a \qquad \forall a \in \mathbb{L}, p \in \mathbb{P} \tag{2.4e}$$

$$y_{a,p} \in \{0,1\} \qquad \forall a \in \mathbb{L}, p \in \mathbb{P} \tag{2.4f}$$

Vemos que la formulación (2.4) es igual que la del problema estático presentada en (1.4), simplemente cambiando la función objetivo según hemos indicado en el Teorema 2.2.7 por un flujo a minimizar. Dada la formulación anterior podemos trabajar con la red y el arco extra considerando $\tau(t,s) = -(H+1)$ y $c(t,s) = \infty$, o bien forzando a que $x_{(t,s)} = v$, donde v es el valor de un flujo máximo estático en la red original. Esto es clave para los siguientes algoritmos que presentaremos.

Dada la similitud de la formulación dinámica y la estática es posible tomar el Algoritmo 5 y, sustituyendo el cálculo de un flujo máximo por un cálculo de un flujo a coste mínimo mediante el Algoritmo 3 en la red modificada, conseguir un algoritmo heurístico para el problema del q-FlowLoc dinámico cuyo coste computacional sería $\mathcal{O}(q \log(q) + q(|\mathbb{L}| + n^2 m))$.

Otra forma de plantear un algoritmo heurístico sería partir del caso 1-FlowLoc dinámico e iterar q veces de manera concreta, aunque veremos que este último tiene una complejidad computacional superior a la del Algoritmo 5 modificado que hemos indicado en el párrafo anterior.

En el Algoritmo 10 hemos tomado el Algoritmo 4 (que es un algoritmo exacto para $q = 1$) para el caso estático y lo hemos adaptado para calcular un flujo a coste mínimo en la red modificada con el arco (t,s). Puesto que en el peor de los casos repetimos el algoritmo para todas las localizaciones de $\mathbb{L}$, y que el Algoritmo 3 para encontrar un flujo a coste mínimo tiene una complejidad computacional de $\mathcal{O}(n^2 m)$, con n el número de nodos y m el de arcos. Se tiene que la complejidad de este algoritmo (exacto) es, en el peor de los casos, $\mathcal{O}(|\mathbb{L}| n^2 m)$.

Algoritmo 10 1-FlowLoc por enumeración dinámico $\mathcal{O}(|\mathbb{L}|n^2m)$

Input: Red de flujo y localización $R_{\mathbb{P}} = ((V, A), (c, \tau, d, m))$, instalación p y horizonte temporal H

Pasos:

1. Añadimos a la red el arco (t, s) extra con $\tau(t, s) = -(H + 1)$ y $c(t, s) = \infty$
2. Fijamos el flujo a coste mínimo $v = \infty$.
3. Tomamos un arco $a \in \mathbb{L}$ no marcado.
4. Si $c(a) \geq d(p)$ entonces actualizamos $c(a) = c(a) - d(p)$.
5. Calculamos el flujo a coste mínimo x^a en la red $R = (V, A \cup \{(t, s)\}, c, \tau)$ con el Algoritmo 3.
6. Marcamos el arco a y reactualizamos la capacidad $c(a) = c(a) + d(p)$.
7. Si $v(x^a) < v$, actualizamos el flujo a coste mínimo $v = v(x^a)$ y la localización $l(p) = a$.
8. Si todos los arcos de $\mathbb{L}$ están marcados, FIN. Si no, volvemos al paso 3.

Plantear una heurística para q instalaciones en el caso dinámico es ahora sencillo, pues basta repetir el Algoritmo 10 q veces, pero se hará de una manera concreta. Para ello, como ya hicimos en el capítulo anterior, ordenaremos las instalaciones por tamaño y, entonces, haremos el bucle para todas las instalaciones.

Algoritmo 11 q-FlowLoc dinámico $\mathcal{O}(q \log(q) + q|\mathbb{L}|n^2m)$

Input: Red de flujo y localización $R_{\mathbb{P}} = ((V, A), (c, \tau, d, m))$, instalaciones $\mathbb{P}$ y horizonte temporal H

Pasos:

1. Definimos $k = 1$. Colocamos $\mathbb{P} = \{p_1, \ldots, p_q\}$ en orden decreciente según su tamaño
$$d(p_1) \geq d(p_2) \geq \cdots \geq d(p_q).$$
2. Obtenemos $l(p_k) = \hat{a}$ la localización óptima de p_k mediante el Algoritmo 10.
3. Fijamos la localización $l(p_{k+i-1}) = \hat{a} \ \ \forall i = 1, \ldots, m(\hat{a})$.
4. Actualizamos $k = k + m(\hat{a})$. Si $k \leq q$ volver al paso 2.
5. Finalizamos el proceso con vector de localizaciones l. FIN.

Si nos fijamos, ordenar un vector de q elementos nos costaría $\mathcal{O}(q \log(q))$. Además, en el bucle, en el peor de los casos, localizaremos solo una instalación por iteración. Por tanto, la complejidad computacional de este algoritmo será $\mathcal{O}(q \log(q) + q|\mathbb{L}|n^2m)$. Como vemos es mayor a la modificación que hemos comentado sobre el Algoritmo 5 con tiempo $\mathcal{O}(q \log(q) + q(|\mathbb{L}| + n^2m))$. El hecho de haberla expuesto aquí es por la gran claridad que presenta en su construcción, sin embargo, para obtener un mejor resultado computacional se recomienda la modificación comentada, pues si el número de posibles localizaciones $\mathbb{L}$ es grande, entonces, ese término dominará la eficiencia de esta última heurística.

3. Evacuación. Caso Práctico

En este capítulo describiremos una técnica muy utilizada en la práctica para mejorar la capacidad de los arcos de una red en una situación real, el llamado *contraflow*. En el caso que analizamos, no resolvemos el problema del contraflow en general, sino que fijamos un sentido a los arcos dependiendo de la escalera de salida más cercana, habiendo realizado un contraflow de forma manual. Si bien es cierto que se hablará sobre él, en general, ya que en los casos de evacuaciones por carretera los arcos (que representan dichas carreteras) tienen una capacidad máxima según el sentido que tengan las carreteras originalmente en la realidad.

Posteriormente, trataremos un caso práctico de evacuación de un centro de enseñanza secundaria, el IES Francisco Salzillo situado en la ciudad de Alcantarilla (Murcia). Intentaremos contrastar la información, que aparece en el plan de autoprotección del centro, utilizando los algoritmos desarrollados en los capítulos precedentes. Además, se considerará la situación ficticia de tener que realizar una exposición fija en el instituto, resolviéndose mediante el modelo q-FlowLoc.

Al lector que quiera profundizar sobre evacuaciones y localizaciones de instalaciones en una red durante emergencias, aplicadas a un caso práctico, se le recomienda leer Kongsomsaksakul et al. (2005); Liu y Yu (2012); Kaisar et al. (2012); Bayram y Yaman (2018); así como Schadschneider et al. (2009), que trata el problema de evacuar una zona mediante un modelo de simulación multiagente, para prever el comportamiento del grupo de personas, aplicado a un avión y un estadio de fútbol.

3.1. Técnica de aumento de flujo por arco. Contraflow

El *contraflow*, o "lane reversal", es una técnica para aumentar artificialmente las capacidades de los arcos en momentos de necesidad (evacuación o emergencia), usando el hecho de que es físicamente posible invertir el sentido de parte del objeto al que representa el arco en la red.

Por ejemplo, en el caso de una evacuación provocada por un huracán, muchos estados de los Estados Unidos tienen las llamadas "hurricane evacuation routes[1]" (ver Figura 3.1), que no es más que una red de carreteras marcadas previamente para las que, en caso de emergencia, se decreta la inversión del sentido. De esta forma, se aumenta la capacidad del flujo que puede viajar en una dirección dentro del hipotético grafo cuyos arcos serían las carreteras.

[1]Para más información: https://www.txdot.gov/driver/weather/hurricane-contraflow-vids.html

Figura 3.1: A la izquierda una señal que representa el sentido de las carreteras colindantes en caso de huracán y a la derecha un carril que se habilita en caso de emergencia que permite la inversión del sentido más adelante

Al lector que quiera conocer más sobre esta técnica y su aplicación práctica se le recomienda el siguiente informe de 2006: `hdl.handle.net/11299/632`. Este es el informe completo de la publicación de Kim y Shekhar (2005). También existen diversos artículos que tratan los modelos que hemos visto en los capítulos anteriores pero con el añadido de usar el contraflow en ellos: Xie et al. (2010); Rebennack et al. (2010); Pyakurel et al. (2014); Pyakurel y Dhamala (2015).

Vamos a poner un ejemplo sencillo para ver el alcance que puede tener esta técnica. En la Figura 3.2 podemos ver como el arco en color verde con una capacidad de 5 se escinde en dos arcos de capacidades 3 y 2. El arco de capacidad de 3 respeta el sentido original, mientras que el arco de capacidad 2 es nuevo y ha invertido el sentido original.

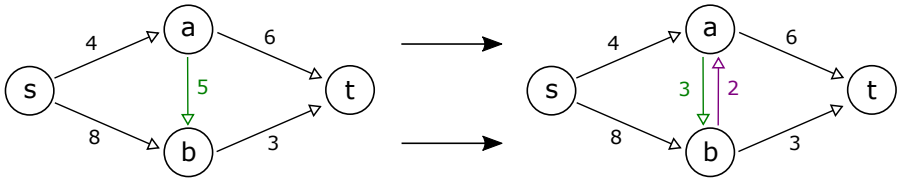

Figura 3.2: Red R transformada por el método del *contraflow*

Esta lógica es aplicable, por ejemplo, a una carretera con 5 carriles todos del mismo sentido. En caso de emergencia, es posible que una parte de los carriles, o incluso la totalidad de ellos, puedan ser usados en el sentido opuesto por los servicios de salvamento. En la Figura 3.2 esto vendría representado por la red de la derecha, en la cual se habrían *habilitado* dos carriles en sentido contrario debido a una situación excepcional de emergencia.

Fijémonos que, en el caso de la red modificada que mostramos en la Figura 3.2, el flujo máximo estático alcanza un valor de 9 unidades, superior al original que era de 7 unidades, gracias a la técnica del *contraflow* aplicada a un solo arco. Esto conlleva que, el tiempo de evacuación necesario para una determinada cantidad de flujo K, sea menor en la red modificada que en la original.

Esta técnica también puede ser utilizada para mejorar flujos dinámicos, abriendo nuevas rutas para el flujo en determinados momentos. Además, si durante la evacuación sucede un imprevisto en algún arco y su capacidad queda disminuida, se puede aplicar el contraflow para invertir los arcos de la red que sean necesarios para intentar recuperar el valor del flujo original.

El problema que, como matemáticos, nos encontramos al tratar la técnica del contraflow es el siguiente y que se plantea en Kim y Shekhar (2005):

Problema del contraflow

Input: Sea $R = ((V, A), (c, \tau))$ una red y x un flujo máximo sobre ella.

Pregunta: ¿Existe una función $f(a) \in \{a^+, a^-\}$ que asigna a cada arco $a \in A$ una dirección (no necesariamente la original) tal que el valor del flujo máximo en la nueva red $R' = (V, f(A))$ sea mayor o igual que $v(x)$?

Este problema hace referencia a encontrar el conjunto óptimo de arcos que deben ser invertidos para maximizar el flujo de una determinada red y que, en particular, harán que el tiempo de evacuación en la red se reduzca desde una cota dada.

Teorema 3.1.1. *El problema del contraflow es $\mathcal{NP}$-completo.*

Demostración. La podemos encontrar en Kim y Shekhar (2005, p. 252) y se basa, al igual que la demostración del Teorema 1.4.3, en una reducción al problema 3-SAT (ver Anexo A.1). □

3.2. Flujo máximo, más rápido y de llegada más temprana en un instituto

Como se ha comentado al inicio del capítulo, aquí trataremos un caso práctico de evacuación de un centro de enseñanza secundaria, el IES Francisco Salzillo, situado en la ciudad de Alcantarilla (Murcia). Para la obtención de los datos reales hemos utilizado el *Plan de Autoprotección 2019-2020*, proporcionado por el Equipo Directivo del centro.

Descripción

El IES Francisco Salzillo cuenta con 3 edificios separados físicamente (ver Figura 3.3). El salón de actos junto con el gimnasio; el edificio principal, con las aulas y departamentos; y un edificio secundario, con únicamente aulas para los alumnos. Al poder tratar estos 3 edificios de manera separada, se pueden plantear evacuaciones independientes. De esta forma, nosotros nos ocuparemos de analizar el caso del edificio principal, que es el más grande y el que puede suponer un mayor reto al tener varias plantas (ver Figura 3.4).

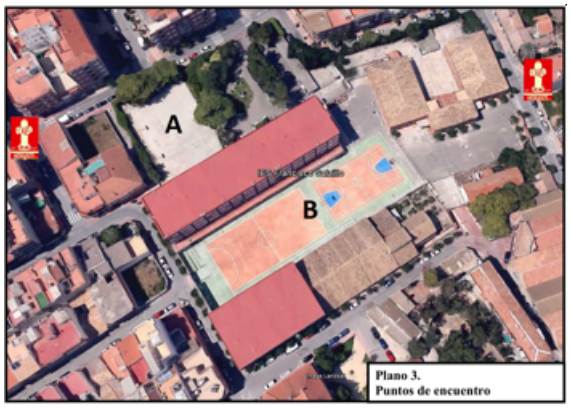

Figura 3.3: Imagen satelital de los tres edificios que conforman el centro IES Francisco Salzillo y sus dos patios, marcados con letras A y B

Todos los años, y siguiendo el plan de autoprotección, se realiza un simulacro de evacuación de todo el centro ante una hipotética situación de incendio en las instalaciones. Dentro del *Plan de Autoprotección 2019-2020*, aparecen especificadas las rutas (en la Figura 3.4) que deben seguir los alumnos y profesorado desde cada estancia hacia la salida más cercana, y el orden en el que deben desalojarse (primero la planta baja y primera, la segunda espera hasta que casi se ha completado la evacuación de la primera y así sucesivamente), siguiendo la premisa de que no se desaloja un aula que esté más lejana de las escaleras sin haber desalojado antes todas las aulas que estén más cerca.

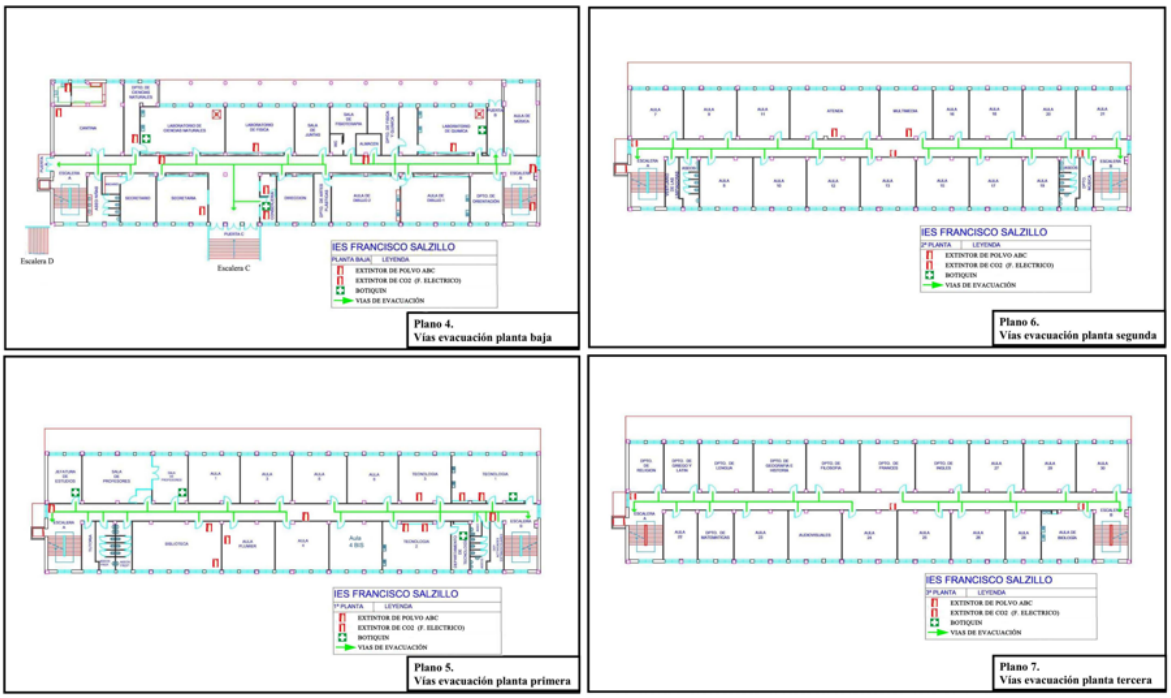

Figura 3.4: Plantas del edificio principal del IES Francisco Salzillo

En el mismo informe, en su página 51, aparece el tiempo estimado para realizar dicho simulacro de evacuación: 2 minutos y 50 segundos. Además del número de individuos que se espera que haya en el edificio (892 personas entre alumnos y profesores). Vamos a modelar con una red el edificio entero y, usando los algoritmos de flujos máximos y flujos más rápidos, ver si dicho tiempo se puede rebajar, e incluso si la forma de evacuar es diferente a la mostrada en la Figura 3.4.

Modelo matemático

Utilizaremos las imágenes de las plantas del edificio principal que aparecen en la Figura 3.4, para construir una red que lo represente. Dicha red R viene dada en las Figuras 3.5, 3.6, 3.7 y 3.8. En ellas se puede ver que cada aula es un nodo con la letra "s", pues todas las aulas son, a priori, fuente de parte del flujo, y se ha considerado que los subíndices complicarían la comprensión de la red más que ayudar. Hemos marcado los pasillos con nodos "p", estos hacen de puntos de control y acumulación de las personas cuando salen de las aulas. Luego tenemos los nodos "e", son los que representan las escaleras y son distinguibles, pues sus capacidades y costes de tránsito son diferentes. Finalmente, los nodos "t" son las salidas del edificio, los destinos del flujo para esta evacuación. Un resumen del número de nodos por planta se tiene en la Tabla 3.1.

Planta 0	$3\times t$ $2\times e$ $11\times s$ $8\times p$	Planta 1	$0\times t$ $4\times e$ $14\times s$ $9\times p$	Planta 2	$0\times t$ $4\times e$ $16\times s$ $9\times p$	Planta 3	$0\times t$ $2\times e$ $19\times s$ $9\times p$

Tabla 3.1: Resumen por plantas de los diferentes nodos de la red

Las capacidades y tiempos, en segundos, que se tarda en atravesar los diferentes arcos aparecen en las Figuras 3.5, 3.6, 3.7 y 3.8 entre paréntesis, $(c(a), \tau(a))$. Además, mostramos al lado de cada nodo, en azul, su capacidad. Es decir, la cantidad de individuos que pueden permanecer a la espera en ese nodo un instante de tiempo. Si sumamos todas las capacidades de todas las aulas del edificio, obtenemos que el número máximo de personas que pueden estar en el edificio simultáneamente es de 1665. Como vemos, este número es superior a las 892 personas que habitualmente ocupan el edificio. Esta diferencia se debe a que muchas aulas (informática, tecnología, dibujo, departamentos, etc.) están vacías o no tienen casi concurrencia. Por ello, como luego describiremos en el análisis de resultados, se añade una variable nueva en el modelo (ver Código A.2) que indica aquellos nodos que, presumiblemente, van a contener el grueso de los individuos (que serán, sobre todo, las plantas segunda y tercera).

Para este caso práctico no se ha resuelto el problema del contraflow en general, sino que hemos fijado un sentido en los arcos dependiendo de la escalera de salida más cercana. Esto tiene sentido mientras todas las vías de escape del instituto estén disponibles desde el inicio. Sin embargo, se puede observar que las aulas situadas en el centro de los pasillos tienen la opción de ir en cualquiera de los dos sentidos, puesto que no podemos distinguir qué sentido es el mejor para esos arcos.

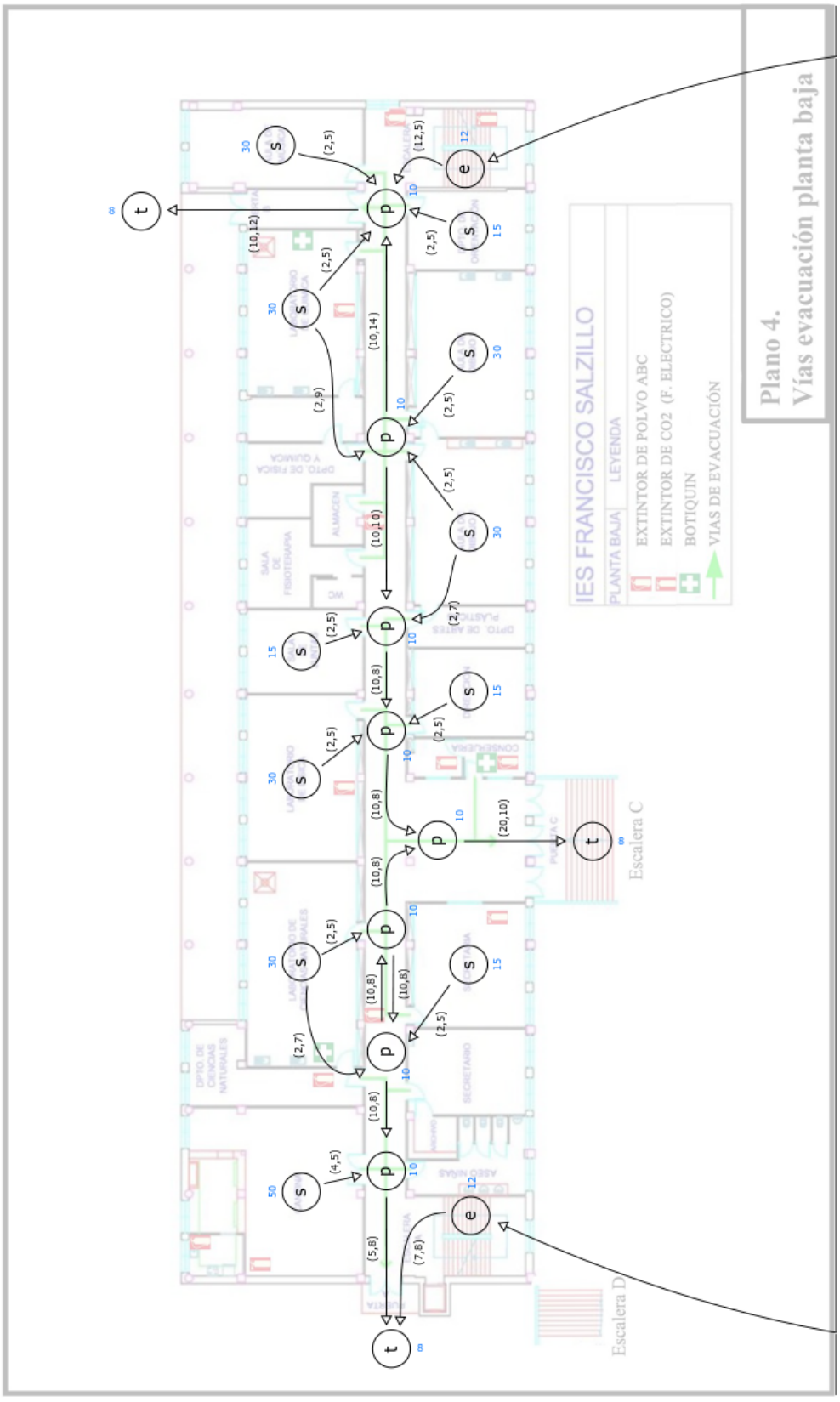

Figura 3.5: Red R que representa la planta baja del IES Francisco Salzillo. Sobre cada arco arco a se muestra $(c(a), \tau(a))$

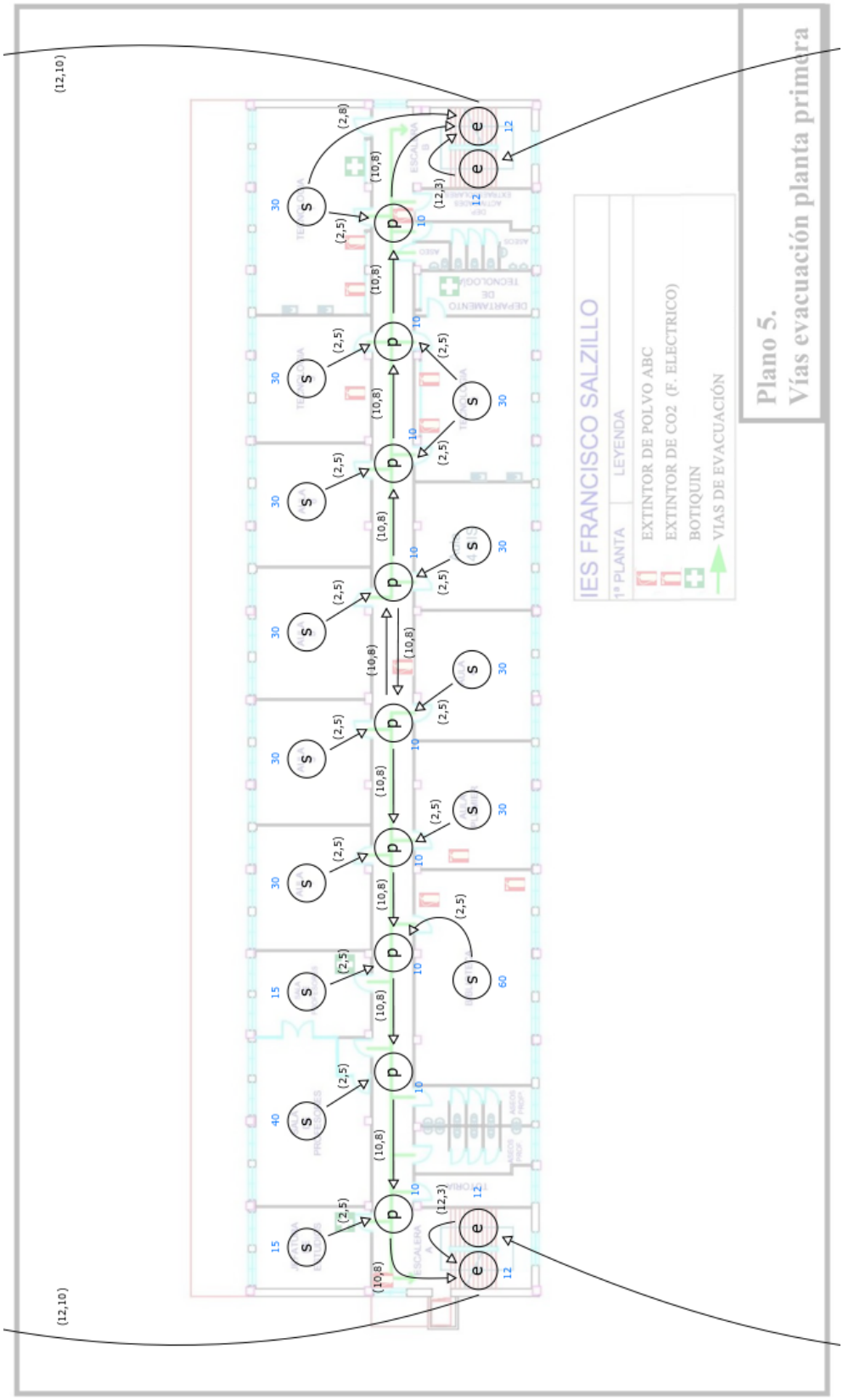

Figura 3.6: Red R que representa la primera planta del IES Francisco Salzillo. Sobre cada arco arco a se muestra $(c(a), \tau(a))$

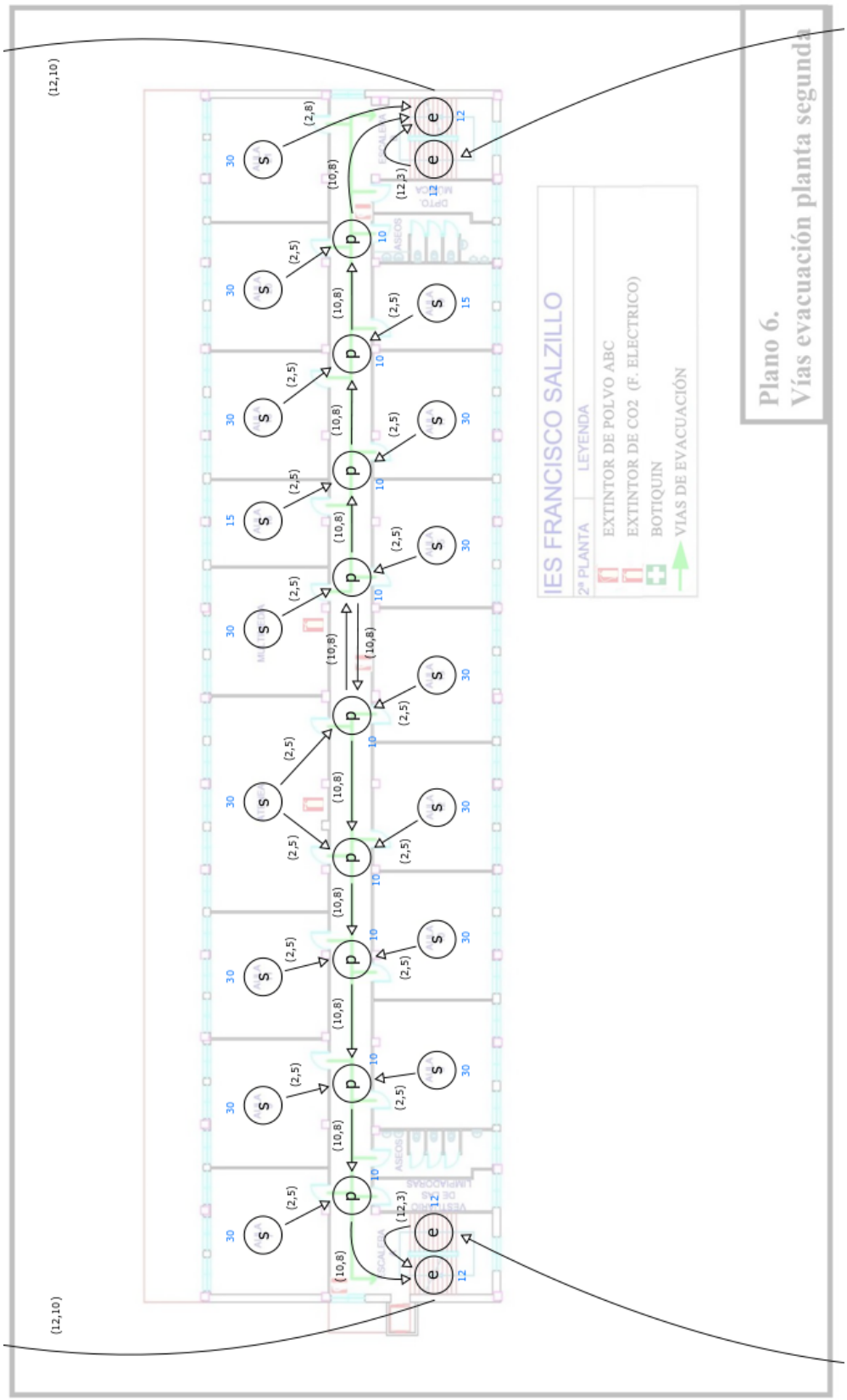

Figura 3.7: Red R que representa la segunda planta del IES Francisco Salzillo. Sobre cada arco arco a se muestra $(c(a), \tau(a))$

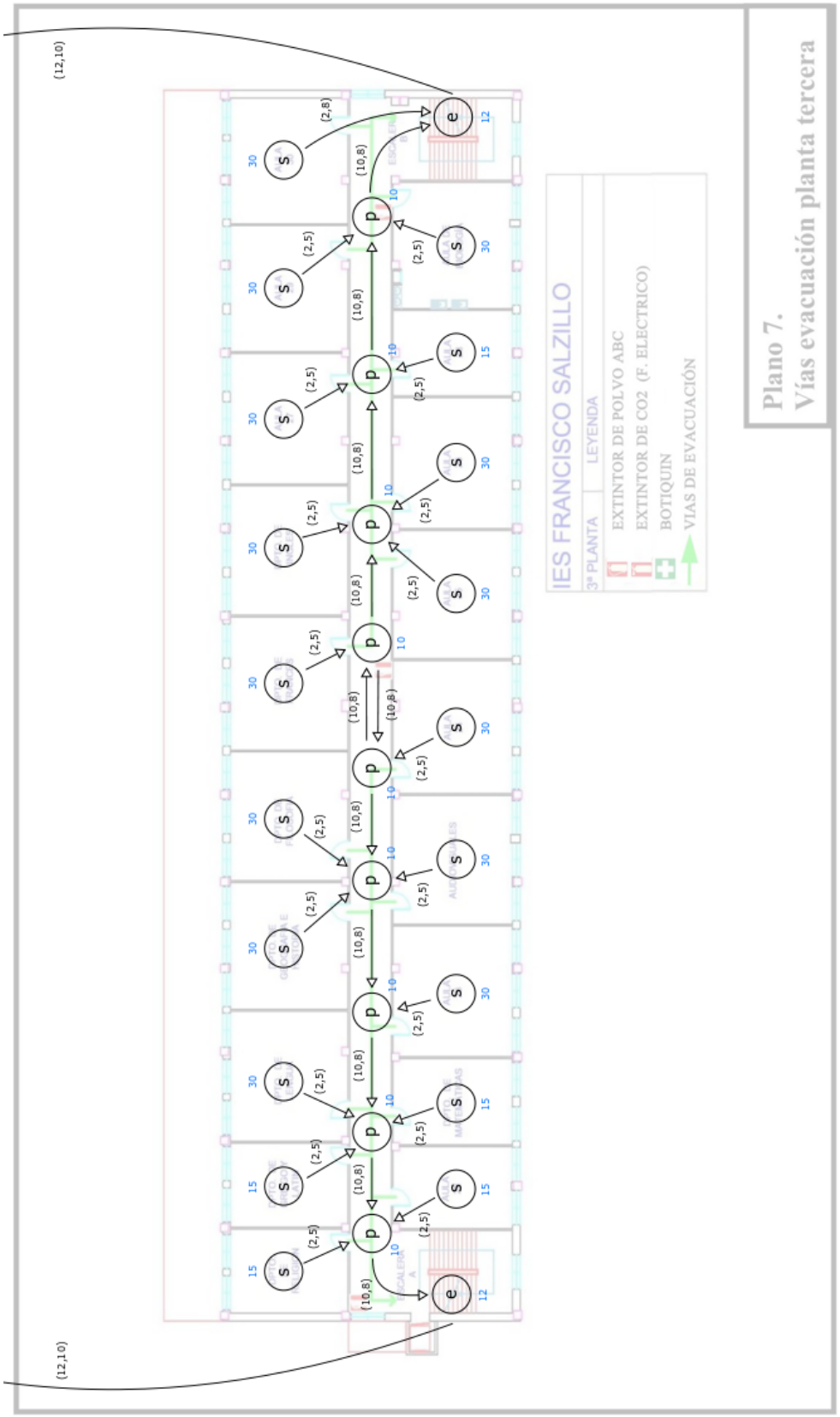

Figura 3.8: Red R que representa la tercera planta del IES Francisco Salzillo. Sobre cada arco arco a se muestra $(c(a), \tau(a))$

Tenemos un modelo con múltiples fuentes y destinos en el que queremos encontrar un flujo máximo **dinámico**. Los algoritmos que hemos desarrollado en el capítulo anterior son todos para redes en las que tenemos una fuente y un destino, es decir, tendremos que realizar una transformación de la red para poder adecuarla a nuestros algoritmos. Como necesitamos recoger el comportamiento conjunto de varias aulas interaccionando, es imperativo mantener varias fuentes de flujo.

Esto lo podemos hacer si tomamos la red R, que representa a todo el edificio, y construimos su red expandida temporalmente hasta un horizonte H. En esta nueva red, R_H, el problema se reduce a encontrar un flujo máximo **estático** con múltiples fuentes y destinos. Por suerte, podemos crear en esta situación dos nodos que aglutinen a todas estas fuentes y destinos, como hicimos en la Figura 2.2 y, con esta modificación, aplicar los resultados del primer capítulo.

Para poder manejar la red R hemos optado por utilizar su matriz de adyacencia, que denotaremos por M. Nos hemos apoyado en la web https://graphonline.ru/es/, que permite introducir un grafo y obtener, entre otras muchas cosas, la matriz de adyacencia asociada[2]. Para el lector que quiera trabajar construyendo grafos, se le recomienda usar la librería *Networkx* de Python.

Como en la Definición 2.1.2 se explicitan las transformaciones que tienen lugar para construir los conjuntos V_H y A_H de la red expandida a partir de la original, podemos crear un código que tome la matriz M y genere la matriz de adyacencia M_H de la red expandida temporalmente. Ya con esta nueva matriz, podemos construir la matriz de incidencia, denotada por B_H, que usaremos para saber cuáles son los arcos sucesores y predecesores de cada uno de los nodos de V_H.

Una vez tenemos todos estos elementos, es posible plantear el problema de programación lineal para calcular el flujo máximo estático en la red R_H (ver la formulación en (1.2)). Este flujo se podrá transformar en un flujo dinámico en la red original R siguiendo la parametrización siguiente:

$$x((i,j),\theta) = x_H\left(i_\theta, j_{\theta+\tau(i,j)}\right) \quad \forall (i,j) \in A,\ i,j \in V,\ \theta \in \{0,\dots,H\}.$$

En general, el flujo máximo dinámico obtenido no será de la familia TRF, ya que trabajamos directamente con la red expandida y no hay obligación de que la solución obtenida repita el mismo trayecto a lo largo del tiempo. Es más, la teoría expuesta en este trabajo no nos asegura que dicho flujo repetido temporalmente pueda existir en una red con múltiples fuentes y destinos.

Código

Hemos optado por programar todo el proceso íntegramente en lenguaje *Mosel* de *Xpress*. Explicaremos aquí las líneas de los dos primeros códigos del Anexo A.2, aunque ya están comentadas en ellos.

2. El grafo que representa al instituto se puede encontrar en: http://graphonline.ru/en/?graph=PLtpFLCFQJypeiTl

Empezamos por el Código A.1. En las primeras líneas, de la 9 a la 12, definimos los dos parámetros iniciales del modelo. El número de nodos de la red original, n, y el horizonte temporal en segundos, H. De las líneas 14 hasta la 38 declaramos los rangos, matrices, conjuntos y variables que vamos a usar en el desarrollo del problema. En concreto, siguiendo la notación del código, el flujo en un arco viene dado por la variable $x(a)$.

Después, entre las líneas 47 y 50, leemos los datos referentes a la red concreta que estamos tratando, desde un archivo externo llamado "Data - IES Salzillo.txt". A continuación, desde la línea 65 hasta la 103, construimos la matriz de adyacencia de la red expandida *MH*, la matriz de costes temporales *tauH* y la de capacidades *cH*; siguiendo las directrices que nos marcaba la Definición 2.1.2 y la observación posterior sobre los nodos "súper s" y "súper t". Estas matrices se construyen siguiendo un patrón de *cajas*, es decir, los primeros $(H+1)\times(H+1)$ elementos de la matriz *MH* provienen del primer elemento de la matriz *M*. Por ello, el tamaño de estas matrices es de $n\cdot(H+1)+2$.

El construir esta matriz (*MH*), aunque para la teoría sea redundante, resulta útil para fijar una numeración en los arcos dentro del mismo código. Esto se hace entre las líneas 107 y 114. Una vez hemos hecho esto, podemos crear la matriz de incidencia de la red expandida, llamada *BH* en el código. Entre las líneas 119 y 122 se rellena esta matriz utilizando la notación siguiente:

- Un elemento (i,a) de esta matriz toma el valor 1 si el arco a es un arco sucesor del nodo i, es decir, $a\in\Gamma(i)$.
- Un elemento (i,a) de esta matriz toma el valor -1 si el arco a es un arco predecesor del nodo i, es decir, $a\in\Gamma^{-1}(i)$.
- Si un elemento (i,a) toma el valor 0, entonces el arco a no incide en el nodo i.

Con esto ya tenemos las matrices necesarias para plantear correctamente el problema de programación lineal descrito en la Formulación (1.2). Desde la línea 133 hasta la 150 se define la función objetivo (el valor de nuestro flujo estático en la red expandida R_H), las restricciones de conservación de flujo, límite de capacidad por arco y no negatividad del flujo. Además, se añade la restricción de integridad para las variables x, pues representan individuos y no tiene sentido una solución fraccionaria. Se han realizado pruebas con la relajación lineal y se han obtenido soluciones enteras sin una mejora sustancial en el tiempo de ejecución.

En la salida del programa, líneas 157 a 159, mostramos por pantalla el valor del flujo máximo estático en la red expandida y el tiempo que ha tardado la máquina en resolver el problema.

El Código A.2 es muy similar al Código A.1, pero se le añade un conjunto extra llamado *AulasLlenas* en la línea 36. Este indicará qué aulas son las que admiten un flujo de individuos y cuáles no (líneas 98 a 102 del código). Se utiliza para sesgar las soluciones del problema hacia un escenario realista en el que los alumnos y profesores se encuentren en las aulas indicadas en el conjunto. También se puede utilizar para crear un escenario tipo que se requiera simular.

Resultados

Fijémonos que el número de nodos de la red R, denotado por n, es 110 y que tiene 122 arcos, denotado por m. Según el informe del *Plan de Autoprotección 2019-2020*, el tiempo que se tarda en realizar la evacuación es de 2 minutos y 50 segundos.

Entonces, el horizonte temporal máximo, en segundos, sobre el que vamos a trabajar es $H = 170$ segundos. Si obtuviésemos que el flujo máximo para ese horizonte temporal no es mayor que las 892 personas que el informe presupone que hay en el edificio, entonces nuestra modelización del edificio estará mal hecha y se tendrán que modificar algunos parámetros de capacidad y/o tiempo para ajustarse más a la realidad.

En la línea 17 del Código A.1, se define una variable que guardará el número de arcos de la red expandida temporalmente, mH. Por la Definición 2.1.2 podemos cuantificarla y acotarla:

$$mH = m \cdot H + m \cdot \sum_{a \in AH} (H - \tau(a) + 1) \leq 2m \cdot H = 41480.$$

Siguiendo la misma definición, podemos cuantificar el número de nodos de la red expandida, que serán $n \cdot (H + 1) + 2$. Por tanto, el tamaño de la matriz de incidencia, BH, será menor o igual que $18812 \cdot 41480 \leq 8 \cdot 10^8$. Esto nos muestra la dificultad computacional a la que uno se puede enfrentar si quiere usar la red expandida temporalmente, y se entiende perfectamente que la teoría de flujos avanzase para resolver estos problemas utilizando otros métodos.

En nuestro caso, *Xpress* permite el trabajo con matrices "sparse" de una manera óptima. Como en la matriz de incidencia, la mayoría de sus elementos son ceros, el problema se hace tratable por computadora. El número de variables de nuestro problema será exactamente mH, puesto que son los arcos de la red.

En primer lugar, no impondremos restricciones en las aulas que queremos que estén vacías o llenas. De esta forma, el flujo empezará desde aquellas aulas que resulten óptimas en cada horizonte temporal. Para ello usaremos el Código A.1 que hemos descrito en la sección anterior. Si resolvemos tres instancias del problema, para $H = 80, 120$ y 170 segundos, vemos en la Tabla 3.2 que el tiempo de ejecución no crece linealmente.

Código A.1	Flujo máx. (personas)	Tiempo de ejecución (segundos)
$n = 110$, $H = 80$	899	27
$n = 110$, $H = 120$	1579	56
$n = 110$, $H = 170$	1665	115

Tabla 3.2: Resumen de tres instancias del Código A.1 lanzadas con los parámetros de la columna de la izquierda

Según la tabla, son suficientes 80 segundos para evacuar a 899 personas, es decir, más de las 892 que fijaba el informe del plan de autoprotección. Si observamos la solución, esto se debe a que sólo se están evacuando las plantas baja y primera, que son las más rápidas de evacuar. Esto no tiene sentido con el uso real que se le da al edificio, puesto que las aulas que están más ocupadas se sitúan en la segunda y tercera planta.

Además, con 170 segundos, se evacúa la totalidad del edificio, pues tiene capacidad para 1665 personas según el modelo que hemos fijado. Es decir, en el tiempo en el que se espera realizar la evacuación real, según el plan de autoprotección, se ha podido evacuar a casi el doble de individuos. Esto puede estar ocurriendo por dos razones principalmente. La primera de ellas es que los tiempos y capacidades que se han considerado en la red R de las Figuras 3.5, 3.6, 3.7 y 3.8 no reflejan la realidad del instituto, o al menos, no reflejan la realidad de una evacuación (en la que hay otros factores que influyen, como el nerviosismo, que hacen que no todo sea perfecto).

La segunda razón nace de que el *Plan de Autoprotección 2019-2020* indica explícitamente que, para la realización de la evacuación, se debe seguir un orden por aulas. Es decir, un aula de un pasillo no empieza a desalojarse hasta que la inmediatamente anterior haya terminado de desalojarse por completo. Esto provoca que los pasillos, que son las vías de escape, no estén al 100% de su capacidad todo el tiempo que dura la evacuación. Por tanto, aquí hay un elemento diferenciador con respecto a la solución a la que se llega (ver Figura 3.9) en la que podemos observar cómo de un aula salen 2 personas al principio, se quedan esperando 13 personas durante unos segundos, para después volver a salir 2 personas y esperar el resto en el aula, y así sucesivamente.

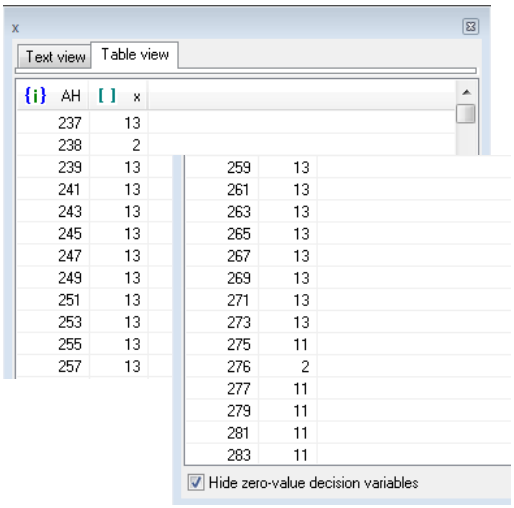

Figura 3.9: Resultado del Código A.2, ($n = 110, H = 120$). El valor de la variable $x(238)$ de la figura es un arco que sale de la clase (sabemos que estos arcos tienen capacidad 2) y el resto de arcos son arcos de espera entre las copias temporales del nodo que representa el aula. No vuelven a salir individuos de la clase hasta el arco 276. Cabe notar que hemos suprimido aquellos valores nulos para una mejor visualización del movimiento del flujo

Esta dinámica "por goteo" es mucho más eficiente que hacer esperar a toda la clase sin realizar ninguna acción, puesto que el pasillo tendrá una tasa de ocupación mayor de esta manera. El problema real que se le plantea a esta solución es que los individuos a evacuar son niños y deben estar controlados por su profesor. Por ello, toda la clase se mueve como un solo impulso de flujo, para poder ser vigilados.

Si queremos realizar una aproximación más realista del problema tenemos que introducir, como se ha dicho antes, el conjunto *AulasLlenas* en el Código A.2. Esto forzará al programa a enviar un flujo a través de los nodos que le fijemos en ese conjunto. En este caso fijaremos las aulas según la ocupación real que creemos que van a tener:

- La planta baja estará vacía ya que las aulas de allí son los laboratorios y las oficinas de registro y gestión del centro.

- La primera planta tendrá 6 aulas con 30 alumnos y la sala de profesores con 15 docentes realizando sus tareas.

- En la segunda planta se consideran las 12 aulas habituales de 30 alumnos llenas, dos aulas pequeñas de 15 alumnos y el aula de informática con otros 30 alumnos en su interior.

- Y en la tercera planta se supondrá que las 8 aulas corrientes que allí se encuentran están con 30 alumnos y que 3 departamentos están ocupados como clases de 15 alumnos.

Esto hace un total de 900 personas en el escenario propuesto, que pensamos, se acerca a una distribución habitual del alumnado. La versatilidad de este conjunto, *AulasLlenas*, reside en que es posible cambiarlo para poder analizar otros escenarios concretos (por ejemplo, laboratorios llenos, solo una planta a máxima capacidad, etc.). En la Tabla 3.3 vemos un resumen de 5 instancias del código lanzadas con el conjunto de aulas que hemos propuesto.

Código A.2	Flujo máx. (personas)	Tiempo de ejecución (segundos)
$n = 110$, $H = 80$	481	27
$n = 110$, $H = 90$	651	32
$n = 110$, $H = 100$	821	38
$n = 110$, $H = 104$	889	41
$n = 110$, $H = 105$	900	41

Tabla 3.3: Resumen de cinco instancias del Código A.2 lanzadas con los parámetros de la columna de la izquierda

Si utilizamos el Lema 2.4.2, podemos calcular el flujo más rápido en la red R utilizando varias instancias repetidas del problema del flujo máximo, hasta encontrar el valor del horizonte temporal H para el cual, en $H - 1$, ya no se evacúa a la cantidad de personas requeridas. En nuestro caso, 892.

Si utilizamos la Tabla 3.2, podemos deducir que el flujo más rápido se tiene con un tiempo menor o igual que 80 segundos. Este caso no es relevante, debido a que la solución se aleja demasiado del escenario habitual en el que se van a encontrar las aulas. En la Tabla 3.3 vemos que:

- Para $H = 104$ segundos se evacúa a 889 personas.

- Para $H = 105$ segundos se evacúa a 900 personas (el máximo admisible).

Por tanto, el flujo más rápido en el segundo caso se tiene con 105 segundos. Dicho de otra forma, según el modelo y aplicando una solución "por goteo" de forma perfecta, se puede reducir en 1 minuto y 5 segundos la estimación para el tiempo de evacuación del IES Francisco Salzillo.

Mejoras a introducir

Hemos encontrado un flujo máximo y un flujo más rápido en la red R. Veamos ahora las dificultades a las que nos enfrentaríamos al tratar de encontrar un flujo de llegada más temprana.

El Algoritmo 7 (Flujo de llegada más temprana) hacía uso de una sucesión de redes residuales para calcular el orden de los caminos de aumento a coste mínimo. Debido a que nuestro modelo ha necesitado varios nodos fuente y destino, hemos tenido que utilizar la red expandida temporalmente. Calcular sobre esta su red residual e ir modificándola según el camino de aumento que se encuentre hasta que no quede ninguno resulta un problema mucho más complicado de lo esperado, computacionalmente hablando.

En el desarrollo del caso práctico, nos hemos ceñido a los algoritmos y heurísticas que hemos expuesto en los capítulos anteriores. Se puede profundizar en el estudio que hemos realizado utilizando el artículo de Baumann y Skutella (2009), donde se muestra un **algoritmo exacto** en tiempo polinomial para el problema del flujo de llegada más temprana en el caso de tener varias fuentes y varios destinos. Se podría adaptar el algoritmo que muestran los autores para un conjunto de clases diferentes, que harán de nodos fuente, y las diferentes salidas del edificio, que harán de nodos destino, pudiendo resolver el problema sin tener que utilizar la red expandida temporalmente como apoyo.

Si aplicásemos este algoritmo junto con el Teorema 2.4.10 (Teorema de triple optimización), se tiene que el resultado es un flujo más rápido desde todas las fuentes a todos los destinos. Si se le imponen restricciones de flujo mínimo en aquellas aulas que se suponga que van a estar ocupadas (como hemos hecho en el Código A.2), obtendremos un flujo de llegada más temprana (y flujo más rápido) que se ajusta a un escenario hipotético más realista.

3.3. Localización de una exposición

Una vez hemos analizado el caso de una evacuación en el IES Francisco Salzillo, propondremos la situación añadida de localizar unas vitrinas para una exposición en los pasillos del instituto. Veremos cómo afecta al flujo máximo y cuáles son las ubicaciones óptimas para un número q de vitrinas dado.

Descripción

Es habitual que los trabajos de los alumnos del Bachillerato Artístico sean expuestos en los pasillos del instituto en vitrinas similares a las de la Figura 3.10. Estas vitrinas limitan la capacidad de los pasillos en 3 personas, la izquierda, y 1 persona, la derecha.

Vamos a suponer que hay que colocar en los pasillos del centro 5 vitrinas de cada tipo. Veamos cómo podemos modelizar esto con la teoría que hemos tratado en los capítulos anteriores y resolvamos el problema del 10-FlowLoc.

Figura 3.10: Ejemplo de vitrinas. A la izquierda, imagen del Museo Arqueológico de Asturias, obtenida online. A la derecha, imagen de una vitrina de IKEA, obtenida online

Modelo matemático

Debido a que el problema del q-FlowLoc es $\mathcal{NP}$-completo (ver Teorema 1.4.3), intentaremos resolver este problema de manera aproximada con la heurística que se propuso en el Algoritmo 5, aplicada a la red expandida temporalmente R_H para un H fijo.

Definiremos un conjunto $\mathbb{P} = \{p_1, \ldots, p_{10}\}$ con las 10 vitrinas, 5 de cada tipo, que hemos indicado antes. Por comodidad, para el razonamiento posterior, se ordenan por tamaño de manera decreciente, quedando en las cinco primeras posiciones las vitrinas con una dimensión de 3 personas y después las de dimensión de 1 persona.

Se considerarán localizaciones admisibles todos los arcos que conectan nodos "p" entre sí y con las escaleras en las Figuras 3.5, 3.6, 3.7 y 3.8. Aunque, para hacerlo más interesante, no se podrán colocar en la planta baja, que está vacía en el escenario analizado anteriormente. Supondremos que solo se puede colocar una vitrina por arco. Esta es una suposición fuerte, pues es posible que en algunos tramos de pasillo se pudiesen colocar dos sin problema alguno. Utilizando la notación del capítulo anterior, sea m_a la capacidad máxima de instalaciones que puede contener el arco a. Entonces, los lugares susceptibles de localización, que representamos por $\mathbb{L}$, son aquellos arcos con $m_a > 0$.

Observemos que habrá que modificar ligeramente el Algoritmo 5, puesto que la red R_H hace referencia a una red expandida, por lo que si una instalación se coloca en un arco, esta debe ser colocada en todas las copias temporales de dicho arco para que sea coherente con el modelo del q-FlowLoc dinámico. Esto se logrará haciendo una numeración de los arcos siguiendo el orden de aparición de las *cajas* de $(H+1) \times (H+1)$ elementos de la matriz MH, de adyacencia de la red expandida, de tal forma que podremos cuantificar en una variable auxiliar el número de copias temporales de cada arco en la red expandida (ver Código A.3, línea 43).

Código

El Código A.3 es el utilizado para resolver el problema de esta sección. Es muy similar a los dos anteriores. La estructura es la misma, pero se le añade un bucle que recorre todas las instalaciones a colocar para replicar la heurística del Algoritmo 5.

De las líneas 9 hasta la 53 se definen los parámetros del modelo y se declaran todas las variables, matrices, conjuntos, etc. que se van a utilizar en el desarrollo del problema. De las líneas 69 a 72 leemos el documento de texto "Data - IES Salzillo (FlowLoc).txt", que contendrá, aparte de las matrices que definen la red, las fuentes y los detinos, el vector de dimensiones de las instalaciones (vitrinas) que se van a colocar ordenado de mayor a menor y una matriz, *maxi*, con las máximas instalaciones que se pueden colocar en cada uno de los arcos de la red R.

Entre las líneas 74 y 164 se crean las matrices de la red expandida, como se hizo en los códigos anteriores. Se numeran en estos bucles los arcos, de tal forma que se consigue una numeración ordenada por las cajas de $(H+1) \times (H+1)$ elementos de MH, que son copias temporales de los elementos de M. Además, se rellena la matriz *num*, definida en la línea 43, que es la que nos permitirá recuperar la relación que existen entre los arcos de la red expandida R_H, que son copias temporales, y los arcos de la red R, utilizando la numeración que nos da el orden de los nodos tal y como aparecen en la matriz de adyacencia M.

Desde la línea 168 hasta la 274 se encuentra el bucle principal del programa. Primero se resuelve el problema de programación lineal original para obtener el flujo máximo sin ninguna instalación (siguiendo el paso 2 del Algoritmo 5). Después, en las líneas 213 a 218, se encuentra el arco elegible para colocar una instalación que, en el óptimo, tenga el mayor espacio libre. Una vez encontrado este arco, de la línea 223 a la 231, se utiliza la matriz *num* para recuperar todas las copias temporales asociadas a este arco y el par de nodos de la red original R (ya que la vitrina es colocada en todas las copias del mismo arco en la red expandida temporalmente).

Con ello, ya podemos actualizar las capacidades de los arcos de la red y comprobar (en la línea 259) si es necesario repetir el problema de programación lineal, pues la instalación estaría afectando al flujo de ese arco (siguiendo el paso 6 del Algoritmo 5). Finalmente, desde la línea 281 hasta la 283 nos ocupamos de la salida del programa.

Resultados

Para plantear la localización de las vitrinas en el instituto se ha considerado el caso más realista que se tiene, es decir, con el conjunto *AulasLlenas* (recordamos que este contiene las aulas que se considera que estarán llenas en el momento de la evacuación).

Se han generado 4 instancias diferentes del problema que listamos en la Tabla 3.4. Si comparamos con los resultados que se obtuvieron en la Tabla 3.3, lo primero que se observa es el incremento en el tiempo de ejecución del programa. Este incremento es debido a que, ahora, realizamos un bucle sobre cada una de las instalaciones y, en el peor de los casos, repetiremos $q = 10$ veces el código referente al problema de programación lineal.

Además, se puede ver que, para el mismo horizonte temporal, el flujo máximo es menor, lo cual tiene sentido pues las vitrinas ocupan un espacio físico que antes era usado para optimizar el flujo.

Código A.3	Flujo máx. (personas)	Tiempo de ejecución (segundos)
$n = 110$, $H = 80$, $q = 10$	461	137
$n = 110$, $H = 90$, $q = 10$	621	197
$n = 110$, $H = 105$, $q = 10$	861	264
$n = 110$, $H = 110$, $q = 10$	900	294

Tabla 3.4: Resumen de cuatro instancias del Código A.3 lanzadas con los parámetros de la columna de la izquierda

Si volvemos a utilizar el Lema 2.4.2, para encontrar un flujo más rápido, vemos que con $H = 110$ ya es suficiente para evacuar a más de 892 personas. Por lo tanto, si lo comparamos con el flujo más rápido de la Tabla 3.3, se puede ver que la colocación de la exposición retrasa solo en 5 segundos el flujo más rápido en la red.

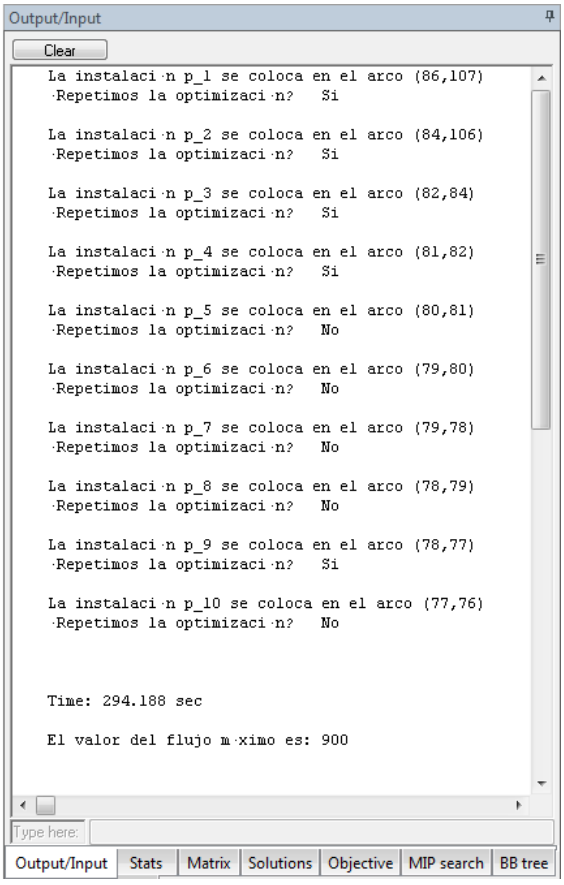

Figura 3.11: Output del Código A.3, ($n = 110, H = 110, q = 10$). Podemos ver el arco, usando la numeración de los nodos marcada por la matriz de adyacencia, en el que se coloca cada vitrina, en el orden dispuesto. Además, vemos si se ha tenido que reoptimizar el problema, debido a la limitación en la capacidad del arco ocupado

Hemos querido hacer la captura del output del programa para el caso del flujo más rápido (ver Figura 3.11). En esta figura se puede observar cómo las primeras 4 vitrinas han necesitado reoptimizar el flujo, ya que la colocación de la vitrina en el arco y, por consiguiente, en todas las copias temporales del mismo, ha disminuido el flujo que pasaba por ese arco en el óptimo. Además, sirve de ejemplo práctico para hacer notar que algunas vitrinas no necesitan reoptimizar el flujo, ya que no ocupan más hueco que el disponible, pero que otras, como la novena vitrina, sí que acaban afectando al flujo (puesto que ya no quedan mejores arcos disponibles).

En la Figura 3.12 representamos la solución del flujo más rápido (instancia $n = 110$, $H = 110$, $q = 10$ en la Tabla 3.4). Se puede observar cómo toda la exposición se coloca entre los pasillos de la primera planta y los huecos de las escaleras. Esto es debido al conjunto *AulasLlenas*, ya que la mayor parte del alumnado se ha concentrado en las plantas segunda y tercera. Como no se ha permitido colocar en la planta baja ninguna vitrina de la exposición, la primera planta es la menos concurrida y disponible que se tiene.

Mejoras a introducir

Como ya se comentó en la sección anterior, con el Código A.3 y el conjunto *AulasLlenas* se puede realizar un **análisis de sensibilidad** de diferentes escenarios de distribución de alumnos en el centro. Esto sería útil para encontrar la mejor posición de las vitrinas en la mayoría de los casos y minimizar el impacto en la evacuación.

En esta sección nos hemos vuelto a circunscribir a la teoría que hemos desarrollado durante los capítulos anteriores. Sin embargo, se recomienda el planteamiento que aparece en Heller y Hamacher (2011). Los autores proponen una heurística, aplicada a un grafo especial (que es una transformación del original), para resolver el problema del q-FlowLoc estático en una red con múltiples fuentes y destinos. Por último, formulan el problema de programación lineal (ver Heller y Hamacher, 2011, p. 524-525) mediante múltiples flujos, que se dirigen desde una fuente a un destino concreto, respetando las capacidades de los arcos y la conservación de cada uno de los flujos. Se maximiza la suma de todos los flujos de la red. Esta formulación se podría adaptar a nuestra red expandida temporalmente, R_H, fijando las copias temporales de los nodos que serán considerados fuentes y destinos.

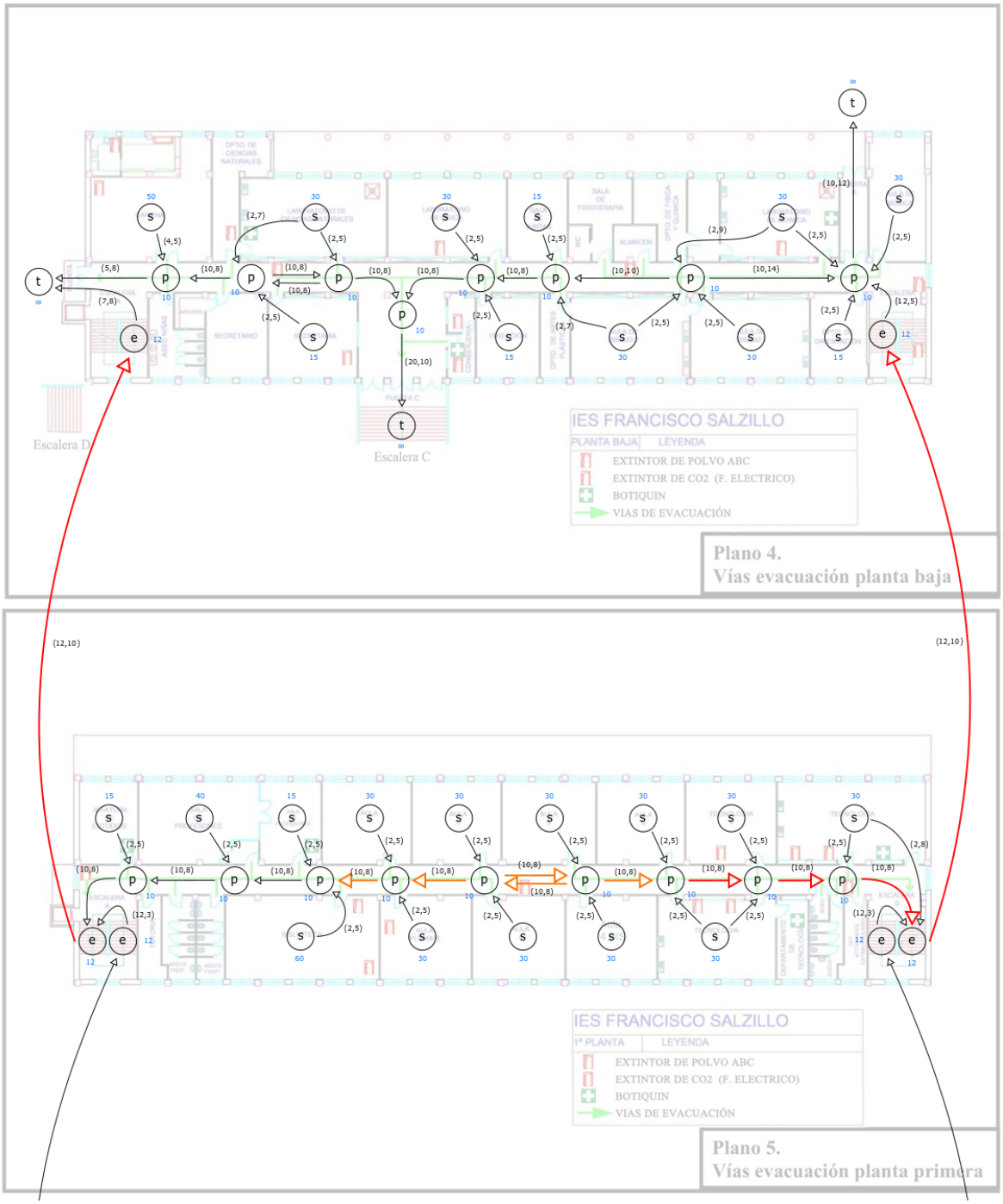

Figura 3.12: Solución del Código A.3, $(n = 110, H = 110, q = 10)$. En los arcos rojos se ubican las vitrinas de tamaño 3 y en los arcos naranjas las vitrinas de tamaño 1

A. Anexos

A.1. Complejidad computacional

Si el lector quisiera profundizar en los temas que vamos a exponer a continuación sobre complejidad computacional le recomendaría las dos lecturas Papadimitriou (1994) y Arora y Barak (2009). Sin embargo, con las pinceladas aquí dadas se espera que se pueda entender su uso durante el trabajo.

Cuando nos referimos a complejidad computacional hablamos de la dificultad inherente de un problema resuelto por ordenador. Más concretamente diremos que dos problemas tienen una complejidad computacional igual si los algoritmos que se usan para resolverlos obtienen sendas soluciones en un tiempo computacional similar. Para medir dicho tiempo se usa la notación de la $\mathcal{O}$ grande.

Definición A.1.1. Dadas dos funciones $f, g : \mathbb{R} \longrightarrow \mathbb{R}$ diremos que $f(x) = \mathcal{O}(g(x))$ si existe $M \in \mathbb{R}^+$ y $x_0 \in \mathbb{R}$ tales que

$$|f(x)| \leq Mg(x) \qquad \forall x \geq x_0.$$

Por tanto, diremos que dos algoritmos tienen la misma complejidad computacional si tienen la misma $\mathcal{O}$, es decir, si sus tiempos computacionales son comparables para casos *grandes* salvo un factor constante.

Tiempo de computación	$x = 10$	$x = 30$	$x = 50$
$\mathcal{O}(log(x))$	0.001 mili seg.	0.0015 mili seg.	0.0017 mili seg.
$\mathcal{O}(x)$	0.01 mili seg.	0.03 mili seg.	0.05 mili seg.
$\mathcal{O}(x^5)$	0.1 seg.	24.3 seg.	5.2 min.
$\mathcal{O}(2^x)$	0.001 seg.	17.9 min.	35.7 años
$\mathcal{O}(3^x)$	0.06 seg.	6.5 años.	22764 mill. años

Tabla A.1: Tiempo de computación si se realizan 10^6 operaciones por segundo

Podemos decir que un algoritmo resuelve un problema en *tiempo polinomial* si $f(x) = \mathcal{O}(x^n)$ con $n \in \mathbb{N}$, en *tiempo exponencial* si $f(x) = \mathcal{O}(a^x)$ con $a \in \mathbb{R}$, en *tiempo logarítmico* si $f(x) = \mathcal{O}(log(x))$ y así con otras funciones que tengan comportamientos diferentes en el infinito.

Como vemos en la Tabla A.1, los algoritmos más *deseables* son aquellos que tienen un tiempo de computación polinomial o logarítmico pues pueden ejecutarse en una cantidad de tiempo admisible. Esto da lugar a la siguiente definición, algo difusa, sobre lo que se considera un problema intratable en la práctica computacional.

Definición A.1.2. Diremos que un problema es **intratable** si es resoluble pero no existe ningún algoritmo eficiente para resolverlo.

Fijémonos que la definición habla sobre lo *eficiente* que es un algoritmo, pero no podemos simplemente decir que un algoritmo con tiempo polinomial es siempre eficiente frente a, por ejemplo, un algoritmo con tiempo exponencial. Así, en la Tabla A.1, vemos que, si el tamaño de un problema es lo suficientemente pequeño, un algoritmo exponencial puede tardar menos tiempo que uno polinomial.

Esto además se enfrenta con el hecho de que el tiempo computacional normalmente se calcula con el peor de los casos a los que se tiene que enfrentar un algoritmo en un problema dado. Por tanto, que el tiempo computacional sea del orden de 2^n quiere decir que en el peor de los casos tardará ese tiempo, pero podría ser que el resto de casos se resolvieran en un tiempo muy inferior.

Ejemplo A.1.3. El algoritmo del símplex para programación lineal se ha demostrado que tiene una complejidad computacional exponencial (ver Klee y Minty, 1972) pero, en la práctica, se obtienen soluciones en un tiempo razonable. Sin embargo, nadie diría que los problemas resueltos con este algoritmo son de facto intratables.

Existe otro grado de complejidad computacional que mezcla la idea de los polinomiales y los exponenciales (o más general, los no polinomiales) y estos son los **pseudopolinomiales**. Se dirá que un algoritmo resuelve un problema en tiempo pesudopolinomial si es un algoritmo polinomial respecto al valor de entrada pero no lo es respecto a la longitud de la entrada.

Ejemplo A.1.4. Supongamos que queremos queremos comprobar cuando un número $n \in \mathbb{N}$ es primo o no. Es conocido el algoritmo en el cual es suficiente comprobar que no existe un divisor de n menor o igual que $\sqrt{n}$ para de deducir que n es primo. Este algoritmo realiza, a lo sumo, $\sqrt{n} - 1$ divisiones y, por tanto, es polinomial (en realidad sublineal) respecto al valor de entrada n. ¿Qué ocurre si vemos las operaciones que debe hacer este algoritmo respecto a la *longitud* del parámetro n?

Si tomamos $n = 10^{100} + 1$ requeriremos, como mucho, 10^{50} divisiones para ver si el número es primo, mientras que el número de dígitos es 100. Esto hace que este algoritmo tenga una complejidad exponencial respecto a la longitud del parámetro de entrada n. Por ello se dice que es *pseudopolinomial*.

Podemos definir unas clases de problemas si atendemos a la Definición A.1.2 sobre lo que entendemos por intratabilidad, computacionalmente hablando.

- $\mathcal{P}$: Problemas que son *resolubles* mediante un algoritmo en tiempo polinomial. Según la tesis de Cobham (ver Bolotin, 2019), un problema es *tratable* solo si está en esta clase. Su razonamiento se apoya en la ambigüedad de la Definición A.1.2 y sabemos que en la actualidad no solo eso los caracteriza.

- $\mathcal{NP}$: Problemas que son *verificables* mediante un algoritmo en tiempo polinomial. Esto significa que, si se tiene un candidato a solución del problema de decisión, entonces se puede comprobar en tiempo polinomial que dicho candidato es solución del problema.

- $\mathcal{NP}$-duro: Problemas que son, al menos, tan difíciles como los problemas más complicados de $\mathcal{NP}$. Algunos no son ni siquiera *decidibles*.

- $\mathcal{NP}$-completo (fuerte): Son los problemas más difíciles contenidos en $\mathcal{NP}$. Esto quiere decir que, si se encuentra un algoritmo en tiempo polinomial que resuelva un problema en esta clase, entonces, todos los problemas de $\mathcal{NP}$ son resolubles en tiempo polinomial (demostrando el famoso Problema del Milenio $\mathcal{P} = \mathcal{NP}$).

- $\mathcal{NP}$-completo (débil): Son los problemas de la clase $\mathcal{NP}$-completo que poseen un algoritmo *pseudopolinomial*.

Una buena referencia para saber más sobre las clases $\mathcal{NP}$ y $\mathcal{NP}$-completo es Garey y Johnson (1979, Cap. 1 y 2), con algunos ejemplos de problemas específicos que están en dichas clases. El honor de ser uno de los primeros problemas en lógica booleana que se demostró que era $\mathcal{NP}$-completo (fuerte) es el problema de la *satisfactibilidad* (SAT). Antes de hablar sobre él tenemos que definir algunos objetos para su entendimiento:

Definición A.1.5. Sea $B = \{b_1, b_2, \dots b_n\}$ un conjunto de variables booleanas una función $t : B \longrightarrow \{T, F\}$ se llama una *asignación verdadera* sobre B.

Diremos, en los términos de la definición, que si $t(b) = T$ entonces b es verdadera y si $t(b) = F$ entonces será falsa. Además, si $b \in B$ es una variable entonces b y $\neg b$ son literales sobre B. Diremos que $\neg b$ será verdadero si y solo si b es falso.

Una cláusula será un conjunto de literales sobre B representando una disyunción ($\vee$), es decir, se *satisface* por una asignación verdadera t si y solo existe al menos un literal que sea verdadero con t. Una colección C de cláusulas sobre B representará la conjunción ($\wedge$) de todas las clausulas que la componen y se dirá que se *satisface* si y solo si existe una asignación verdadera sobre B para todas las cláusulas simultáneamente.

Ejemplo A.1.6. Tomemos $B = \{b_1, b_2\}$ y $C = \{\{b_1, \bar{b_2}\}, \{\bar{b_1}, b_2\}\} = (b_1 \vee \neg b_2) \wedge (\neg b_1 \vee b_2)$, entonces, una asignación verdadera sería $t(b_1) = t(b_2) = T$, por lo que C si que se satisface. Sin embargo $C' = (b_1 \vee b_2) \wedge (b_1 \vee \neg b_2) \wedge \neg b_1$ no se satisface nunca.

SATISFACTIBILIDAD (SAT)

Input: Dado un conjunto B de variables booleanas y una colección C de cláusulas sobre B.

Pregunta: ¿Existe una *asignación verdadera* para C?

Teorema A.1.7 (Teorema de Cook)**.** *El problema SAT definido en el párrafo anterior es $\mathcal{NP}$-completo.*

Demostración. Se puede encontrar en Garey y Johnson (1979, p.38-44). □

Una vez definido el problema SAT podemos tomar una versión restringida del mismo que, en la literatura sobre complejidad, se suele usar como base para probar que otros problemas son $\mathcal{NP}$-completos a partir de este pues su estructura, más definida que la del SAT, hace que trabajar con la colección C de cláusulas sea más cómodo.

3-SATISFACTIBILIDAD (3-SAT)

Input: Dado un conjunto B de variables booleanas y una colección C de cláusulas sobre B con exactamente 3 literales.

Pregunta: ¿Existe una *asignación verdadera* para C?

A partir del Teorema de Cook y transformando el SAT en un problema 3-SAT se tiene el siguiente teorema.

Teorema A.1.8. *El problema 3-SAT es $\mathcal{NP}$-completo.*

Demostración. La demostración se puede encontrar en Garey y Johnson (1979, p.48-49). □

A.2. Códigos

Código A.1: FlujoMax - IES Salzillo

```
model SalzilloFlujoMax
uses "mmxprs";
uses "mmsystem";

!-------------------------------------------------------------------
! Declaraciones iniciales.
!-------------------------------------------------------------------

parameters
    n = 110       ! Nº de nodos de la red.
    H = 170       ! Horizonte temporal (en segundos).
end-parameters

declarations
    V = 1..n      ! Conjunto de nodos de la red.
    VH: range     ! Conjunto de nodos de la red expandida. Tamaño (n*(H+1))+2.
    mH: integer   ! Nº de arcos de la red expandida.
    AH: range     ! Conjunto de arcos de la red expandida. Tamaño mH.

    M: array(V, V) of integer  ! Matriz de adyacencia de la red.
    c: array(V, V) of integer  ! Matriz de capacidades de la red.
    tau: array(V, V) of integer  ! Matriz de costes temp. de la red.

    MH: array(VH, VH) of integer  ! Matriz de adyacencia de la red expandida.
    cH: array(VH, VH) of integer  ! Matriz de capacidades de la red expandida.
    tauH: array(VH, VH) of integer ! Matriz de costes temp. de la red expandida.

    cAH: array(AH) of integer  ! Vector de capacidades de la red expandida.
    tauAH: array(AH) of integer  ! Vector de costes temp. de la red expandida.

    numH: array(VH, VH) of integer ! Matriz auxiliar con la numeración de los arcos.
    BH: array(VH, AH) of integer  ! Matriz de incidencia de la red expandida.

    S: set of integer    ! Conjunto de nodos fuente/source.
    T: set of integer    ! Conjunto de nodos destino/sink.

    x: array(AH) of mpvar    ! Variables, ="flujo en el arco a de AH".
end-declarations

starttime:= gettime

! Inicializamos las matrices originales de adyacencia, costes y capacidades
!  en los arcos a partir de un archivo externo construido previamente.
!  Además, destacamos el conjunto de nodos fuente y destino para que
!  el código sepa dirigir el flujo correctamente.

initializations from 'Data - IES Salzillo.txt'
    M; c; tau;
    S; T;
end-initializations
```

```
60
61 ! Generamos, a partir de la matriz de adyacencia original, la matriz
62 !  de adyacencia de la red expandida temporalmente y, con ella,
63 !  la matriz de capacidades y costes temporales.
64
65 forall(i,  j in V)  do
66     if(M(i, j) = 0)  then
67         if(i= j)  then
68             forall(theta in 1..H)  do
69                 MH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ 1)  := 1
70                 cH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ 1)  := c(i, j)
71                 tauH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ 1)  := 1
72             end-do
73         end-if
74     else
75         ! Si M(i,j)=1 entonces (i_theta, j_{theta+tau(i,j)}) = 1; cero en el resto.
76         forall(theta in 1..(H-tau(i, j) + 1) )  do
77             MH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) )  := 1
78             cH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) )  := c(i, j)
79             tauH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) )  := tau(i, j)
80         end-do
81     end-if
82 end-do
83
84
85 ! Rellenamos las dos últimas columnas y filas de las matrices
86 !  (corresponden a los nodos agregados super s y super t).
87
88 forall(i in V,  theta in 1..H+ 1 | i in T)  do
89     MH((i-1) *(H+ 1) + theta, (n*(H+ 1) ) + 2)  := 1
90     cH((i-1) *(H+ 1) + theta, (n*(H+ 1) ) + 2)  := 2147483647! ~infinito
91     tauH((i-1) *(H+ 1) + theta, (n*(H+ 1) ) + 2)  := 0! theta, para flujo ponderado.
92 end-do
93
94 forall(j in V | j in S)  do
95     MH((n*(H+ 1) ) + 1, (j-1) *(H+ 1) + 1)  := 1
96     cH((n*(H+ 1) ) + 1, (j-1) *(H+ 1) + 1)  := c(j, j)
97     tauH((n*(H+ 1) ) + 1, (j-1) *(H+ 1) + 1)  := 0
98 end-do
99
100 ! Datos del arco (t,s).
101 MH((n*(H+ 1) ) + 2, (n*(H+ 1) ) + 1)  := 1
102 cH((n*(H+ 1) ) + 2, (n*(H+ 1) ) + 1)  := 2147483647! ~infinito
103 tauH((n*(H+ 1) ) + 2, (n*(H+ 1) ) + 1)  := -(H+ 1)
104
105 ! Vamos a numerar los arcos, según el orden de aparición en la matriz MH.
106 !  El valor final es -> mH := sum(i in VH, j in VH) MH(i,j) <= 2*m*H
107 mH := 0
108
109 forall(i,  j in VH | MH(i, j)  = 1) do
110     numH(i, j)  := mH+ 1
111     cAH(numH(i, j) )  := cH(i, j)
112     tauAH(numH(i, j) )  := tauH(i, j)
113     mH + = 1
114 end-do
115
116 ! Damos el valor de 1 a los arcos sucesores de los nodos y -1 para los
117 !  arcos predecesores. Esto nos servirá en la formulación.
118
119 forall(i,  j in VH | numH(i, j)  <>0) do
120     BH(i, numH(i, j) )  := 1
121     BH(j, numH(i, j) )  := -1
122 end-do
123
```

```
124
125 ! Creamos las variables de decisión x(a).
126 forall(a in AH)  create(x(a) )
127
128
129 !-----------------------------------------------------------------------
130 ! Definimos nuestro modelo lineal, función objetivo y restricciones.
131 !-----------------------------------------------------------------------
132
133 ! Valor del flujo estático.
134 objetivo := x(mH)
135
136 ! Restricciones para la conservación de flujo en la red.
137 forall(i in VH)  sum(a in AH| BH(i, a)  = 1) x(a)  = sum(a in AH| BH(i, a)  = -1) x(a)
138
139 ! Restricciones de flujo menor que la capacidad.
140 forall(a in AH)  x(a)  <= cAH(a)
141
142 ! Restricciones de flujo no negativo.
143 forall(a in AH)  x(a)  >= 0
144
145 ! Restricciones de integridad pues son personas.
146 forall(a in AH)  x(a)  is_integer
147
148
149 setparam("XPRS_VERBOSE", 1)
150 maximize(objetivo)
151
152
153 !-----------------------------------------------------------------------
154 ! Output de nuestro modelo.
155 !-----------------------------------------------------------------------
156
157 writeln("")
158 writeln("Time: ", gettime-starttime, " sec\n")
159 writeln("El valor del flujo máximo es: ", getsol(objetivo) , "\n")
160
161 end-model
```

Código A.2: FlujoMax - IES Salzillo + AulasLlenas

```
model SalzilloFlujoMax_AulasLlenas
uses "mmxprs";
uses "mmsystem";

!-------------------------------------------------------------------
! Declaraciones iniciales.
!-------------------------------------------------------------------

parameters
    n = 110       ! Nº de nodos de la red.
    H = 170       ! Horizonte temporal (en segundos).
end-parameters

declarations
    V = 1..n      ! Conjunto de nodos de la red.
    VH: range     ! Conjunto de nodos de la red expandida. Tamaño (n*(H+1))+2.
    mH: integer   ! Nº de arcos de la red expandida.
    AH: range     ! Conjunto de arcos de la red expandida. Tamaño mH.

    M: array(V, V) of integer  ! Matriz de adyacencia de la red.
    c: array(V, V) of integer  ! Matriz de capacidades de la red.
    tau: array(V, V) of integer  ! Matriz de costes temp. de la red.

    MH: array(VH, VH) of integer  ! Matriz de adyacencia de la red expandida.
    cH: array(VH, VH) of integer  ! Matriz de capacidades de la red expandida.
    tauH: array(VH, VH) of integer ! Matriz de costes temp. de la red expandida.

    cAH: array(AH) of integer  ! Vector de capacidades de la red expandida.
    tauAH: array(AH) of integer  ! Vector de costes temp. de la red expandida.

    numH: array(VH, VH) of integer ! Matriz auxiliar con la numeración de los arcos.
    BH: array(VH, AH) of integer  ! Matriz de incidencia de la red expandida.

    S: set of integer   ! Conjunto de nodos fuente/source.
    T: set of integer   ! Conjunto de nodos destino/sink.
    AulasLlenas: set of integer  ! Conjunto de nodos por los que permitimos pasar el flujo.

    x: array(AH) of mpvar   ! Variables, ="flujo en el arco a de AH".
end-declarations

starttime:= gettime

! Inicializamos las matrices originales de adyacencia, costes y capacidades
!  en los arcos a partir de un archivo externo construido previamente.
!  Además, destacamos el conjunto de nodos fuente y destino para que
!  el código sepa dirigir el flujo correctamente.

initializations from 'Data - IES Salzillo.txt'
    M; c; tau;
    S; T; AulasLlenas;
end-initializations
```

```
64
65 ! Generamos, a partir de la matriz de adyacencia original, la matriz
66 !  de adyacencia de la red expandida temporalmente y, con ella,
67 !  la matriz de capacidades y costes temporales.
68
69 forall(i, j in V) do
70     if(M(i, j) = 0) then
71         if(i= j) then
72             forall(theta in 1..H) do
73                 MH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ 1) := 1
74                 cH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ 1) := c(i, j)
75                 tauH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ 1) := 1
76             end-do
77         end-if
78     else
79         ! Si M(i,j)=1 entonces (i_theta, j_{theta+tau(i,j)}) = 1; cero en el resto.
80         forall(theta in 1..(H-tau(i, j) + 1) ) do
81             MH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) ) := 1
82             cH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) ) := c(i, j)
83             tauH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) ) := tau(i, j)
84         end-do
85     end-if
86 end-do
87
88
89 ! Rellenamos las dos últimas columnas y filas de las matrices
90 !  (corresponden a los nodos agregados super s y super t).
91
92 forall(i in V, theta in 1..H+ 1 | i in T) do
93     MH((i-1) *(H+ 1) + theta, (n*(H+ 1) ) + 2) := 1
94     cH((i-1) *(H+ 1) + theta, (n*(H+ 1) ) + 2) := 2147483647! ~infinito
95     tauH((i-1) *(H+ 1) + theta, (n*(H+ 1) ) + 2) := 0! theta, para flujo ponderado.
96 end-do
97
98 forall(j in V | j in S and j in AulasLlenas) do
99     MH((n*(H+ 1) ) + 1, (j-1) *(H+ 1) + 1) := 1
100    cH((n*(H+ 1) ) + 1, (j-1) *(H+ 1) + 1) := c(j, j)
101    tauH((n*(H+ 1) ) + 1, (j-1) *(H+ 1) + 1) := 0
102 end-do
103
104 ! Datos del arco (t,s).
105 MH((n*(H+ 1) ) + 2, (n*(H+ 1) ) + 1) := 1
106 cH((n*(H+ 1) ) + 2, (n*(H+ 1) ) + 1) := 2147483647! ~infinito
107 tauH((n*(H+ 1) ) + 2, (n*(H+ 1) ) + 1) := -(H+ 1)
108
109 ! Vamos a numerar los arcos, según el orden de aparición en la matriz MH.
110 !  El valor final es -> mH := sum(i in VH, j in VH) MH(i,j) <= 2*m*H
111 mH := 0
112
113 forall(i, j in VH | MH(i, j) = 1) do
114     numH(i, j) := mH+ 1
115     cAH(numH(i, j) ) := cH(i, j)
116     tauAH(numH(i, j) ) := tauH(i, j)
117     mH += 1
118 end-do
119
120 ! Damos el valor de 1 a los arcos sucesores de los nodos y -1 para los
121 !  arcos predecesores. Esto nos servirá en la formulación.
122
123 forall(i, j in VH | numH(i, j) <>0) do
124     BH(i, numH(i, j) ) := 1
125     BH(j, numH(i, j) ) := -1
126 end-do
127
```

```
128
129 ! Creamos las variables de decisión x(a).
130 forall(a in AH)  create(x(a) )
131
132
133 !-----------------------------------------------------------------------
134 ! Definimos nuestro modelo lineal, función objetivo y restricciones.
135 !-----------------------------------------------------------------------
136
137 ! Valor del flujo estático.
138 objetivo := x(mH)
139
140 ! Restricciones para la conservación de flujo en la red.
141 forall(i in VH)  sum(a in AH| BH(i, a)  = 1) x(a)  = sum(a in AH| BH(i, a)  = -1) x(a)
142
143 ! Restricciones de flujo menor que la capacidad.
144 forall(a in AH)  x(a)  <= cAH(a)
145
146 ! Restricciones de flujo no negativo.
147 forall(a in AH)  x(a)  >= 0
148
149 ! Restricciones de integridad pues son personas.
150 forall(a in AH)  x(a)  is_integer
151
152
153 setparam("XPRS_VERBOSE", 1)
154 maximize(objetivo)
155
156
157 !-----------------------------------------------------------------------
158 ! Output de nuestro modelo.
159 !-----------------------------------------------------------------------
160
161 writeln("")
162 writeln("Time: ", gettime-starttime, " sec\n")
163 writeln("El valor del flujo máximo es: ", getsol(objetivo) , "\n")
164
165 end-model
```

Código A.3: FlujoMax y FlowLoc - IES Salzillo + AulasLlenas

```
model SalzilloFlowLoc_AulasLlenas
uses "mmxprs";
uses "mmsystem";

!-------------------------------------------------------------------
! Declaraciones iniciales.
!-------------------------------------------------------------------

parameters
    n = 110      ! Nº de nodos de la red.
    q =  10      ! Nº de instalaciones a colocar en la red.
    H = 110      ! Horizonte temporal (en segundos).
end-parameters

declarations
    V = 1..n     ! Conjunto de nodos de la red.
    P = 1..q     ! Conjunto de instalaciones a colocar en la red.
    VH: range    ! Conjunto de nodos de la red expandida. Tamaño (n*(H+1))+2.
    mH: integer  ! Nº de arcos de la red expandida.
    AH: range    ! Conjunto de arcos de la red expandida. Tamaño mH.

    M: array(V, V) of integer  ! Matriz de adyacencia de la red.
    c: array(V, V) of integer  ! Matriz de capacidades de la red.
    tau: array(V, V) of integer  ! Matriz de costes temp. de la red.
    d: array(P) of integer  ! Tamaño/dimensión de cada instalación.
    maxi: array(V, V) of integer  ! Matriz del máx. nº de instalaciones que se pueden colocar en la red.
    L: array(V, V) of integer  ! Matriz del lugar que ocupa una cierta instalación.

    MH: array(VH, VH) of integer  ! Matriz de adyacencia de la red expandida.
    cH: array(VH, VH) of integer  ! Matriz de capacidades de la red expandida.
    tauH: array(VH, VH) of integer ! Matriz de costes temp. de la red expandida.
    maxiH: array(VH, VH) of integer ! Matriz del máx. nº de instalaciones que se pueden colocar en R_H.
    LH: array(VH, VH) of integer  ! Matriz del lugar que ocupa una cierta instalación en la red expandida.

    cAH: array(AH) of integer  ! Vector de capacidades de la red expandida.
    tauAH: array(AH) of integer  ! Vector de costes temp. de la red expandida.
    maxiAH: array(AH) of integer ! Vector del máx. nº de instalaciones que se pueden colocar en R_H.
    LAH: array(AH) of integer  ! Vector del lugar que ocupa una cierta instalación en la red expandida.

    numH: array(VH, VH) of integer ! Matriz auxiliar con la numeración de los arcos.
    BH: array(VH, AH) of integer  ! Matriz de incidencia de la red expandida.

    num: array(range, 1..3) of integer  ! Matriz auxiliar que nos proporciona el arco (i,j) de R y sus ↩
        ↪ correspondientes copias temporales en R_H.
    contador: integer    ! Contador auxiliar para num.
    repetirOptimizacion: boolean ! Booleano para comprobar si repetimos el problema de optimización.


    S: set of integer    ! Conjunto de nodos fuente/source.
    T: set of integer    ! Conjunto de nodos destino/sink.
    AulasLlenas: set of integer  ! Conjunto de nodos por los que permitimos pasar el flujo.

    x: array(AH) of mpvar    ! Variables, ="flujo en el arco a de AH".
end-declarations

starttime:= gettime
repetirOptimizacion := true
```

```
63
64 ! Inicializamos las matrices originales de adyacencia, costes y capacidades
65 !  en los arcos a partir de un archivo externo construido previamente.
66 !  Además, destacamos el conjunto de nodos fuente y destino para que
67 !  el código sepa dirigir el flujo correctamente.
68
69 initializations from 'Data - IES Salzillo (FlowLoc).txt'
70     M; c; tau; d; maxi;
71     S; T; AulasLlenas;
72 end-initializations
73
74 mH:= 0 !  El valor final es -> mH := sum(i in VH, j in VH) MH(i,j)
75 contador:= 0
76 num(contador+ 1, 1)  := 1;
77 contador + = 1
78
79
80 ! Generamos, a partir de la matriz de adyacencia original, la matriz
81 !  de adyacencia de la red expandida temporalmente y con ella
82 !  la matriz de capacidades y costes temporales.
83
84 forall(i,  j in V)  do
85     if(M(i, j) = 0)  then
86         if(i= j)  then
87             forall(theta in 1..H)  do
88                 MH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ 1)  := 1
89                 cH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ 1)  := c(i, j)
90                 tauH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ 1)  := 1
91                 numH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ 1)  := mH+ 1
92                 mH + = 1
93             end-do
94             num(contador+ 1, 1)  := mH+ 1; num(contador, 2)  := i; num(contador, 3)  := j
95             contador + = 1
96         end-if
97     else
98         ! Si M(i,j)=1 entonces (i_theta, j_{theta+tau_ij}) = 1; cero en el resto.
99         forall(theta in 1..(H-tau(i, j) + 1) )  do
100             MH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) )  := 1
101             cH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) )  := c(i, j)
102             tauH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) )  := tau(i, j)
103             maxiH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) )  := maxi(i, j)
104             numH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) )  := mH+ 1
105             mH + = 1
106         end-do
107         num(contador+ 1, 1)  := mH+ 1; num(contador, 2)  := i; num(contador, 3)  := j
108         contador + = 1
109     end-if
110 end-do
111
112
113 ! Rellenamos las dos últimas columnas y filas de las matrices
114 !  (corresponden a los nodos agregados super s y super t).
115
116 forall(i in V,  theta in 1..H+ 1 | i in T)  do
117     MH((i-1) *(H+ 1) + theta, (n*(H+ 1) ) + 2)  := 1
118     cH((i-1) *(H+ 1) + theta, (n*(H+ 1) ) + 2)  := 2147483647! ~infinito
119     tauH((i-1) *(H+ 1) + theta, (n*(H+ 1) ) + 2)  := 0! theta, para flujo ponderado.
120     numH((i-1) *(H+ 1) + theta, (n*(H+ 1) ) + 2)  := mH+ 1
121     mH + = 1
122 end-do
123 num(contador+ 1, 1)  := mH+ 1; num(contador, 2)  := -1; num(contador, 3)  := n+ 2
124 contador + = 1
125
126
```

```
127
128 forall(j in V | j in S) do
129     if(j in AulasLlenas) then
130         MH((n*(H+ 1) ) + 1, (j-1) *(H+ 1) + 1) := 1
131         cH((n*(H+ 1) ) + 1, (j-1) *(H+ 1) + 1) := c(j, j)
132         tauH((n*(H+ 1) ) + 1, (j-1) *(H+ 1) + 1) := 0
133         numH((n*(H+ 1) ) + 1, (j-1) *(H+ 1) + 1) := mH+ 1
134         mH += 1
135     end-if
136 end-do
137 num(contador+ 1, 1) := mH+ 1; num(contador, 2) := n+ 1; num(contador, 3) := -1
138 contador += 1
139
140 ! Datos del arco (t,s).
141 MH((n*(H+ 1) ) + 2, (n*(H+ 1) ) + 1) := 1
142 cH((n*(H+ 1) ) + 2, (n*(H+ 1) ) + 1) := 2147483647! ~infinito
143 tauH((n*(H+ 1) ) + 2, (n*(H+ 1) ) + 1) := -(H+ 1)
144 numH((n*(H+ 1) ) + 2, (n*(H+ 1) ) + 1) := mH+ 1
145 mH += 1
146 num(contador+ 1, 1) := mH+ 1; num(contador, 2) := n+ 2; num(contador, 3) := n+ 1
147 contador += 1
148
149
150 ! Construimos los vectores de datos siguiendo el orden de numH por cajas en MH.
151
152 forall(i, j in VH | MH(i, j) = 1) do
153     cAH(numH(i, j) ) := cH(i, j)
154     tauAH(numH(i, j) ) := tauH(i, j)
155     maxiAH(numH(i, j) ) := maxiH(i, j)
156 end-do
157
158 ! Damos el valor de 1 a los arcos sucesores de los nodos y -1 para los
159 ! arcos predecesores. Esto nos servirá en la formulación.
160
161 forall(i, j in VH | numH(i, j) <>0) do
162     BH(i, numH(i, j) ) := 1
163     BH(j, numH(i, j) ) := -1
164 end-do
165
166 ! Bucle principal del programa:
167
168 forall(p in P) do
169
170     ! Creamos las variables de decisión x(a).
171     forall(a in AH) create(x(a) )
172
173     if(repetirOptimizacion) then
174
175         !-----------------------------------------------------------
176         ! Definimos nuestro modelo lineal, función objetivo y restricciones.
177         !-----------------------------------------------------------
178
179         ! Valor del flujo estático.
180         objetivo := x(mH)
181
182         ! Restricciones para la conservación de flujo en la red.
183         forall(i in VH) sum(a in AH| BH(i, a) = 1) x(a) = sum(a in AH| BH(i, a) = -1) x(a)
184
185         ! Restricciones de flujo menor que la capacidad.
186         forall(a in AH) x(a) <= cAH(a)
187
188         ! Restricciones de flujo no negativo.
189         forall(a in AH) x(a) >= 0
190
```

```
191         ! Restricciones de integridad pues son personas.
192         forall(a in AH)  x(a)  is_integer
193
194
195         !setparam("XPRS_VERBOSE",1)
196         maximize(objetivo)
197
198     end-if
199
200     !-------------------------------------------------------------------
201     ! Preparación de la localización óptima.
202     !-------------------------------------------------------------------
203
204     ! Reiniciamos os parámetros del bucle.
205     aux := 0.
206     agorro := 0
207     repetirOptimizacion := false
208
209
210     ! Buscamos el arco en R_H que tenga el mayor de los espacios disponibles.
211     !  agorro pertenece a argmax{cAH(a)-x(a)}
212
213     forall(a in AH | maxiAH(a)  >0) do
214         if(aux <cAH(a) -getsol(x(a) ) )  then
215             agorro := a
216         end-if
217         aux := cAH(a) -getsol(x(a) )
218     end-do
219
220     ! Buscamos el arco en la red R que corresponda a agorro.
221     !  Localizamos la instalación p en ese arco de R.
222
223     forall(e in 1..contador | num(e, 1)  >agorro)  do
224         L(num(e-1, 2) , num(e-1, 3) )  := p
225         rango_agorro := num(e-1, 1) ..(num(e, 1) -1)
226
227         ! Output de la heurística de colocación.
228         writeln("La instalación p_", p, " se coloca en el arco (", num(e-1, 2) , ",", num(e-1, 3) , ")")
229
230         break
231     end-do
232
233     ! Actualizamos todas las matrices de localización con la nueva instalación.
234
235     forall(i, j in V | M(i, j)  = 1and L(i, j)  <>0) do
236         forall(theta in 1..(H-tau(i, j) + 1) )  do
237             LH((i-1) *(H+ 1) + theta, (j-1) *(H+ 1) + theta+ tau(i, j) )  := L(i, j)
238         end-do
239     end-do
240
241     forall(i,  j in VH | numH(i, j)  <>0and LH(i, j)  <>0) do
242         LAH(numH(i, j) )  := LH(i, j)
243     end-do
244
245     ! Actualizamos TODAS las copias temporales del arco ocupado con
246     !  la instalación p-ésima.
247
248     forall(a in rango_agorro | LAH(a)  <>0) do
249         if(cAH(a) - d(p)  >0) then
250             cAH(a)  := cAH(a) - d(p)
251         else
252             cAH(a)  := 0
253         end-if
254
```

```
255
256         ! Comprobamos si la instalación afecta al flujo actual.
257         !  Si lo hace, repetimos la optimización.
258
259         if(getsol(x(a) ) >cAH(a) )  repetirOptimizacion := true
260
261         ! Descontamos un espacio disponible en todos los arcos.
262         maxiAH(a)  -= 1
263     end-do
264
265
266     ! Output de la heurística de colocación.
267
268     if(repetirOptimizacion)  then
269         writeln("¿Repetimos la optimización? Sí \n")
270     else
271         writeln("¿Repetimos la optimización? No \n")
272     end-if
273
274 end-do
275
276
277 !-------------------------------------------------------------------
278 ! Output de nuestro modelo.
279 !-------------------------------------------------------------------
280
281 writeln("\n")
282 writeln("Time: ", gettime-starttime, " sec\n")
283 writeln("El valor del flujo máximo es: ", getsol(objetivo) , "\n")
284
285 end-model
```

Bibliografía

Ahuja, R., Magnanti, T. L., y Orlin, J. B. (1993). *Network flows: Theory, algorithms and applications*. (**ISBN-10:** 0-13-617549-X) [1, 2, 9]

Anderson, E. J., Nash, P., y Philpott, A. B. (1982). A class of continuous network flow problems. *Mathematics of Operations Research*, *7*(4), 501–514. Descargado de https://doi.org/10.1287/moor.7.4.501 [36]

Arora, S., y Barak, B. (2009). *Computational complexity: A modern approach*. Cambridge University Press. (**ISBN-10:** 0-521-42426-7) [85]

Azaron, A., y Kianfar, F. (2003). Dynamic shortest path in stochastic dynamic networks: Ship routing problem. *European Journal of Operational Research*, *144*(1), 138–156. Descargado de https://doi.org/10.1016/S0377-2217(01)00385-X [30]

Baumann, N., y Skutella, M. (2009). Earliest arrival flows with multiple sources. *Mathematics of Operations Research*, *34*(2), 499–512. Descargado de https://doi.org/10.1287/moor.1090.0382 [47, 79]

Bayram, V., y Yaman, H. (2018). A stochastic programming approach for shelter location and evacuation planning. *RAIRO-Operations Research*, *52*(3), 779–805. Descargado de https://doi.org/10.1051/ro/2017046 [65]

Bolotin, A. (2019). Cobham's thesis as the sorites paradox. *Journal of Advances in Mathematics and Computer Science*, 1–6. Descargado de https://doi.org/10.9734/JAMCS/2019/46241 [87]

Bookbinder, J. H., y Sethi, S. P. (1980). The dynamic transportation problem: A survey. *Naval Research Logistics Quarterly*, *27*(1), 65–87. Descargado de https://doi.org/10.1002/nav.3800270107 [30]

Burkard, R. E., Dlaska, K., y Klinz, B. (1993). The quickest flow problem. *Zeitschrift für Operations Research*, *37*(1), 31–58. Descargado de https://doi.org/10.1007/BF01415527 [III, 52, 57]

Busacker, R. G., y Gowen, P. J. (1961). A procedure for determining a family of minimum-cost flow patterns. *Operations Research Office Technical Report*, *15*. Descargado de https://apps.dtic.mil/dtic/tr/fulltext/u2/249662.pdf [I, 12]

Çalık, H., Labbé, M., y Yaman, H. (2019). p-center problems. En *Location science* (pp. 51–65). Springer. Descargado de https://doi.org/10.1007/978-3-030-32177-2_3 (**ISBN-13:** 978-3-030-32176-5) [13]

Chalmet, L. G., Francis, R. L., y Saunders, P. B. (1982). Network models for building evacuation. *Fire Technology*, *18*(1), 90–113. Descargado de https://doi.org/10.1007/BF02993491 [26]

Chen, Y., y Chin, Y. (1990). The quickest path problem. *Computers & Operations Research*, *17*(2), 153–161. Descargado de https://doi.org/10.1016/0305-0548(90)90039-A [47]

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269–271. Descargado de https://doi.org/10.1007/BF01386390 [13, 28]

Fleischer, L., y Skutella, M. (2003). Minimum cost flows over time without intermediate storage. En *Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 66–75). (**ISBN-10:** 0-89871-538-5) [29]

Fleischer, L., y Tardos, É. (1998). Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, *23*(3-5), 71–80. Descargado de https://doi.org/10.1016/S0167-6377(98)00037-6 [57]

Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, *5*(6), 345. Descargado de https://doi.org/10.1145/367766.368168 [13]

Ford Jr, L. R., y Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, *8*, 399–404. Descargado de https://doi.org/10.4153/CJM-1956-045-5 [I, 9]

Ford Jr, L. R., y Fulkerson, D. R. (1958). Constructing maximal dynamic flows from static flows. *Operations Research*, *6*(3), 419–433. Descargado de https://doi.org/10.1287/opre.6.3.419 [II, 23, 31, 32, 39, 49]

Ford Jr, L. R., y Fulkerson, D. R. (1962). *Flows in networks*. Princeton University Press. (**ISBN-10:** 0-691-07962-5) [I, 1, 10, 31]

Gale, D. (1959). Transient flows in networks. *Michigan Math. J.*, *6*(1), 59–63. Descargado de https://doi.org/10.1307/mmj/1028998140 [39]

Garey, M. R., y Johnson, D. S. (1979). *Computers and intractability* (Vol. 174). Freeman San Francisco. (**ISBN-10:** 0-7167-1044-7) [87, 88]

Goerigk, M., Hamacher, H. W., y Kinscherff, A. (2018). Ranking robustness and its application to evacuation planning. *European Journal of Operational Research*, *264*(3), 837–846. Descargado de https://doi.org/10.1016/j.ejor.2016.05.037 [30]

Goldberg, A. V., y Tarjan, R. E. (1987). Solving minimum-cost flow problems by successive approximation. En *Nineteenth Annual ACM Symposium on the Theory of Computing* (pp. 7–18). Descargado de https://doi.org/10.1145/28395.28397 (**ISBN-10:** 0-89791-221-7) [13]

Goldberg, A. V., y Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *Journal of the ACM (JACM)*, *35*(4), 921–940. Descargado de https://doi.org/10.1145/48014.61051 [4]

Groß, M., y Skutella, M. (2011). Generalized maximum flows over time. En *International Workshop on Approximation and Online Algorithms* (pp. 247–260). Descargado de https://doi.org/10.1007/978-3-642-29116-6_21 (**ISBN-13:** 978-3-642-29115-9) [30]

Hamacher, H. W., Heller, S., y Rupp, B. (2013). Flow location (flowloc) problems: dynamic network flows and location models for evacuation planning. *Annals of Operations Research*, *207*(1), 161–180. Descargado de https://doi.org/10.1007/s10479-011-0953-9 [II, III, 1, 17, 21]

Heller, S., y Hamacher, H. W. (2011). The multi terminal q-flowloc problem: A heuristic. En *Network Optimization - 5th International Conference* (pp. 523–528). Descargado de https://doi.org/10.1007/978-3-642-21527-8_57 (**ISBN-13:** 978-3-642-21526-1) [14, 83]

Hoppe, B., y Tardos, É. (1994). Polynomial time algorithms for some evacuation problems. En *Fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (Vol. 94, pp. 433–441). (**ISBN-10:** 0-89871-329-3) [47]

Jarvis, J. J., y Ratliff, H. D. (1982). Some equivalent objectives for dynamic network flow problems. *Management science*, *28*(1), 106–109. Descargado de https://doi.org/10.1287/mnsc.28.1.106 [III, 57]

Kaisar, E. I., Hess, L., Palomo, P., y Benazir, A. (2012). An emergency evacuation planning model for special needs populations using public transit systems. *Journal of Public Transportation*, *15*(2), 3. Descargado de http://doi.org/10.5038/2375-0901.15.2.3 [65]

Kim, S., y Shekhar, S. (2005). Contraflow network reconfiguration for evacuation planning: A summary of results. En C. Shahabi y O. Boucelma (Eds.), *Thirteenth Annual ACM International Workshop on Geographic Information Systems* (pp. 250–259). Descargado de https://doi.org/10.1145/1097064.1097099 (**ISBN-10:** 1-59593-146-5) [66, 67]

Klee, V., y Minty, G. J. (1972). How good is the simplex algorithm? *Inequalities*, *3*(3), 159–175. Descargado de https://mathscinet.ams.org/mathscinet-getitem?mr=0332165 [86]

Kongsomsaksakul, S., Yang, C., y Chen, A. (2005). Shelter location-allocation model for flood evacuation planning. *Journal of the Eastern Asia Society for Transportation Studies*, *6*, 4237–4252. Descargado de https://doi.org/10.11175/easts.6.4237 [65]

Liu, Y., y Yu, J. (2012). Emergency evacuation planning for highly populated urban zones: A transit-based solution and optimal operational strategies. En B. Eksioglu (Ed.), *Emergency Management* (cap. 4). Rijeka: IntechOpen. Descargado de https://doi.org/10.5772/33653 (**ISBN-13:** 978-953-307-989-9) [65]

Marín, A., y Pelegrín, M. (2019). p-median problems. En *Location science* (pp. 25–50). Springer. Descargado de https://doi.org/10.1007/978-3-030-32177-2_2 (**ISBN-13:** 978-3-030-32176-5) [13]

Minieka, E. (1973). Maximal, lexicographic, and dynamic network flows. *Operations Research*, *21*(2), 517–527. Descargado de https://doi.org/10.1287/opre.21.2.517 [III, 42]

Moore, E. F. (1959). The shortest path through a maze. En *International Symposium on the Theory of Switching, Part II* (pp. 285–292). [13]

Orda, A., y Rom, R. (1995). On continuous network flows. *Operations Research Letters*, *17*(1), 27–36. Descargado de https://doi.org/10.1016/0167-6377(94)00035-5 [30]

Orlin, J. B. (1983). Maximum-throughput dynamic network flows. *Mathematical Programming*, *27*(2), 214–231. Descargado de https://doi.org/10.1007/BF02591946 [29]

Papadimitriou, C. M. (1994). *Computational complexity*. Reading, Massachusetts: Addison-Wesley. (**ISBN-10:** 0-201-53082-1) [85]

Pelegrín, B., Cánovas, L., y Fernández, P. (1992). *Algoritmos en grafos y redes*. PPU. (**ISBN-10:** 84-477-0031-3) [1, 3, 12, 13]

Pyakurel, U., y Dhamala, T. N. (2015). Models and algorithms on contraflow evacuation planning network problems. *International Journal of Operations Research*, *12*(2), 36–46. Descargado de http://www.orstw.org.tw/ijor/vol12no2/IJOR2015_vol12_no2_p036_p046_final.pdf [66]

Pyakurel, U., Hamacher, H. W., y Dhamala, T. N. (2014). Generalized maximum dynamic contraflow on lossy network. *International Journal of Operations Research Nepal*, *3*(1), 27–44. Descargado de https://n9.cl/zq5x [66]

Rebennack, S., Arulselvan, A., Elefteriadou, L., y Pardalos, P. M. (2010). Complexity analysis for maximum flow problems with arc reversals. *Journal of Combinatorial Optimization*, *19*(2), 200–216. Descargado de https://doi.org/10.1007/s10878-008-9175-8 [66]

Schadschneider, A., Klüpfel, H., Kretz, T., Rogsch, C., y Seyfried, A. (2009). Fundamentals of pedestrian and evacuation dynamics. En A. L. Bazzan y F. Klügl (Eds.), *Multi-Agent Systems for Traffic and Transportation Engineering* (pp. 124–154). IGI Global. Descargado de https://doi.org/10.4018/978-1-60566-226-8.ch006 (**ISBN-13:** 978-1-60566-226-8) [65]

Skutella, M. (2009). An introduction to network flows over time. En W. J. Cook, L. Lovász, y J. Vygen (Eds.), *Research Trends in Combinatorial Optimization* (pp. 451–482). Springer. Descargado de https://doi.org/10.1007/978-3-540-76796-1_21 (**ISBN-13:** 978-3-540-76795-4) [24, 29]

Tardos, E., y Wayne, K. D. (1998). Simple generalized maximum flow algorithms. En *International Conference on Integer Programming and Combinatorial Optimization* (pp. 310–324). Descargado de https://doi.org/10.1007/3-540-69346-7_24 (**ISBN-10:** 3-540-64590-X) [30]

Wilkinson, W. L. (1971). An algorithm for universal maximal dynamic flows in a network. *Operations Research*, *19*(7), 1602–1612. Descargado de https://doi.org/10.1287/opre.19.7.1602 [42]

Xie, C., Lin, D.-Y., y Waller, S. T. (2010). A dynamic evacuation network optimization problem with lane reversal and crossing elimination strategies. *Transportation research part E: Logistics and transportation review*, *46*(3), 295–316. Descargado de https://doi.org/10.1016/j.tre.2009.11.004 [66]

Índice terminológico

P

R

T

V