Flujo máximo

Román Castellarin
casi calco exacta de las diapos de Ariel Zylber

UPC Training Camp
original: Training Camp Argentina

Septiembre 2019

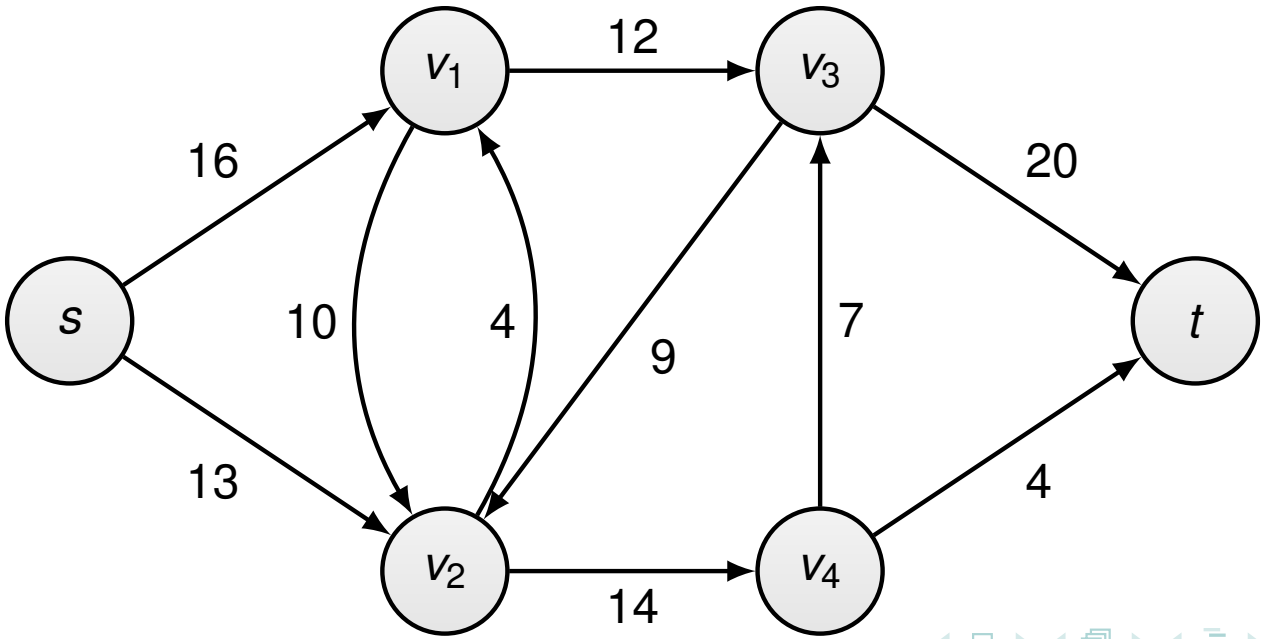
Redes de flujo

Una red de flujo consiste en:

- Un grafo dirigido G con pesos en las aristas.
- Dos vértices distinguidos s (la fuente) y t (el sumidero).

Nuestro objetivo es mandar la mayor cantidad de unidades de flujo de s a t.

- los pesos de las aristas representan la cantidad máxima de flujo que soportan.



Redes de flujo

Por cuestiones de simpleza [1], se podría pedir adicionalmente que:

- No haya bucles (podemos ignorarlos).
- No haya arcos paralelos o antiparalelos.
- No haya arcos entrantes a s.
- No haya arcos salientes de t.

[1] algunas implementaciones de los algoritmos se rompen si alguna de estas condiciones no se satisfacen

Flujo en redes

Un flujo sobre una red es una asignación de un real no negativo a cada arco tal que:

- Por cada arco digo cuantas unidades de flujo pasan por él.
- No le asigno a una arco más de su capacidad máxima.
- En cada vértice (salvo s y t) el flujo que entra tiene que ser igual al que sale.

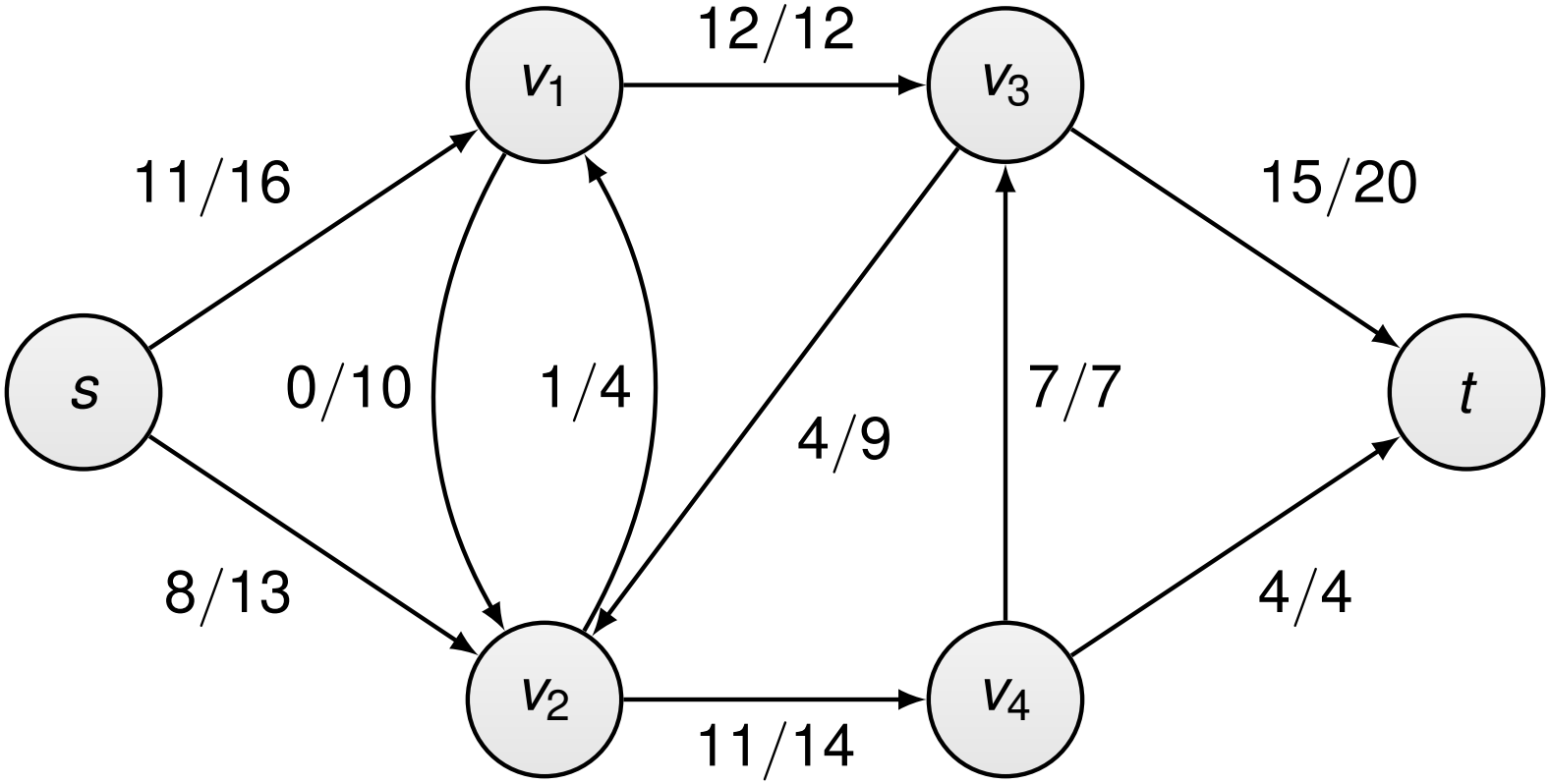
El valor del flujo es:

- La cantidad de flujo que sale de s.
- La cantidad de flujo que llega a t.
- Ambos números siempre coinciden.

Queremos encontrar cual es el mayor valor posible de un flujo sobre la red.

Ejemplo

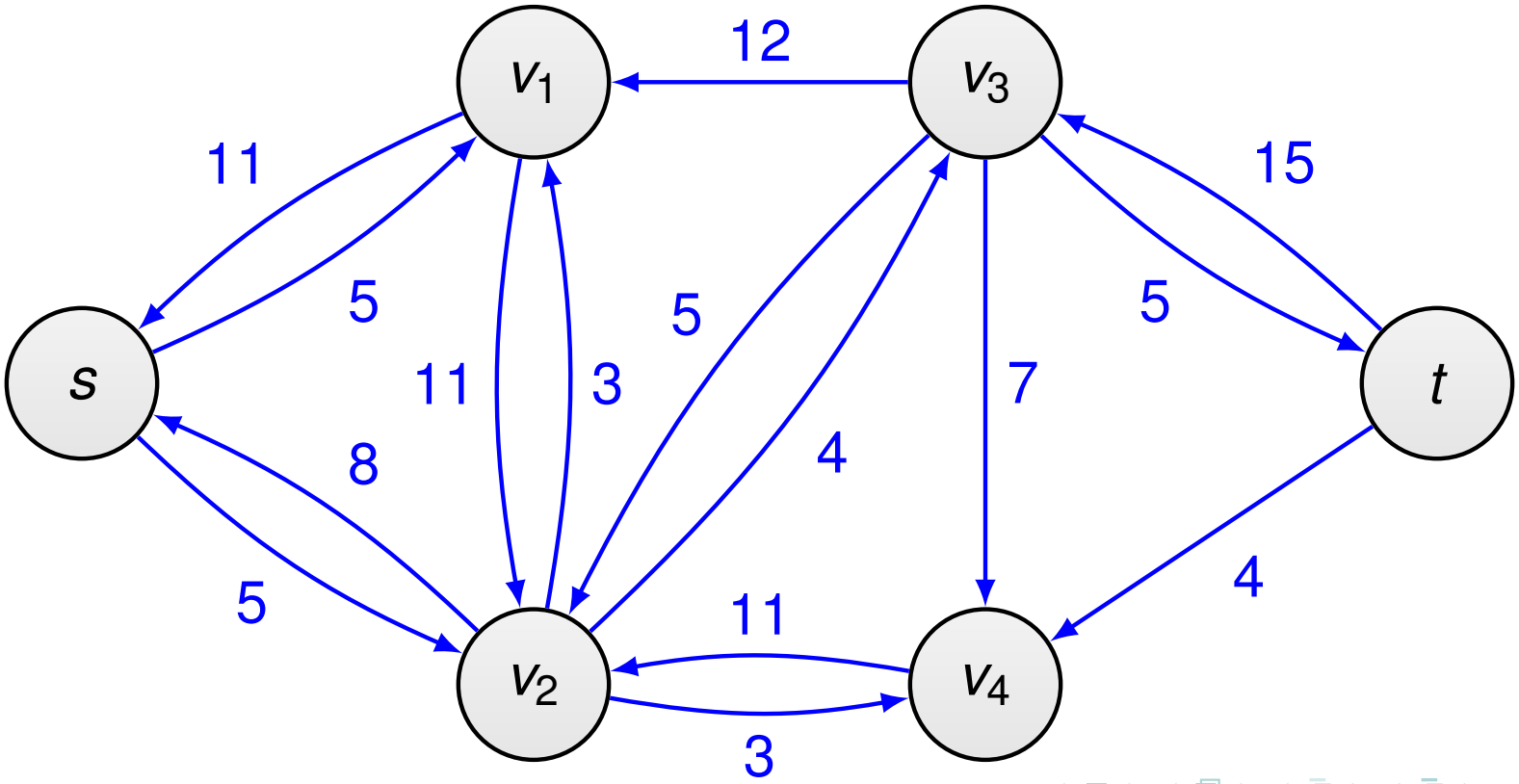
Este es un posible flujo sobre la red:



Red residual

La red residual de un flujo es un grafo dirigido tal que:

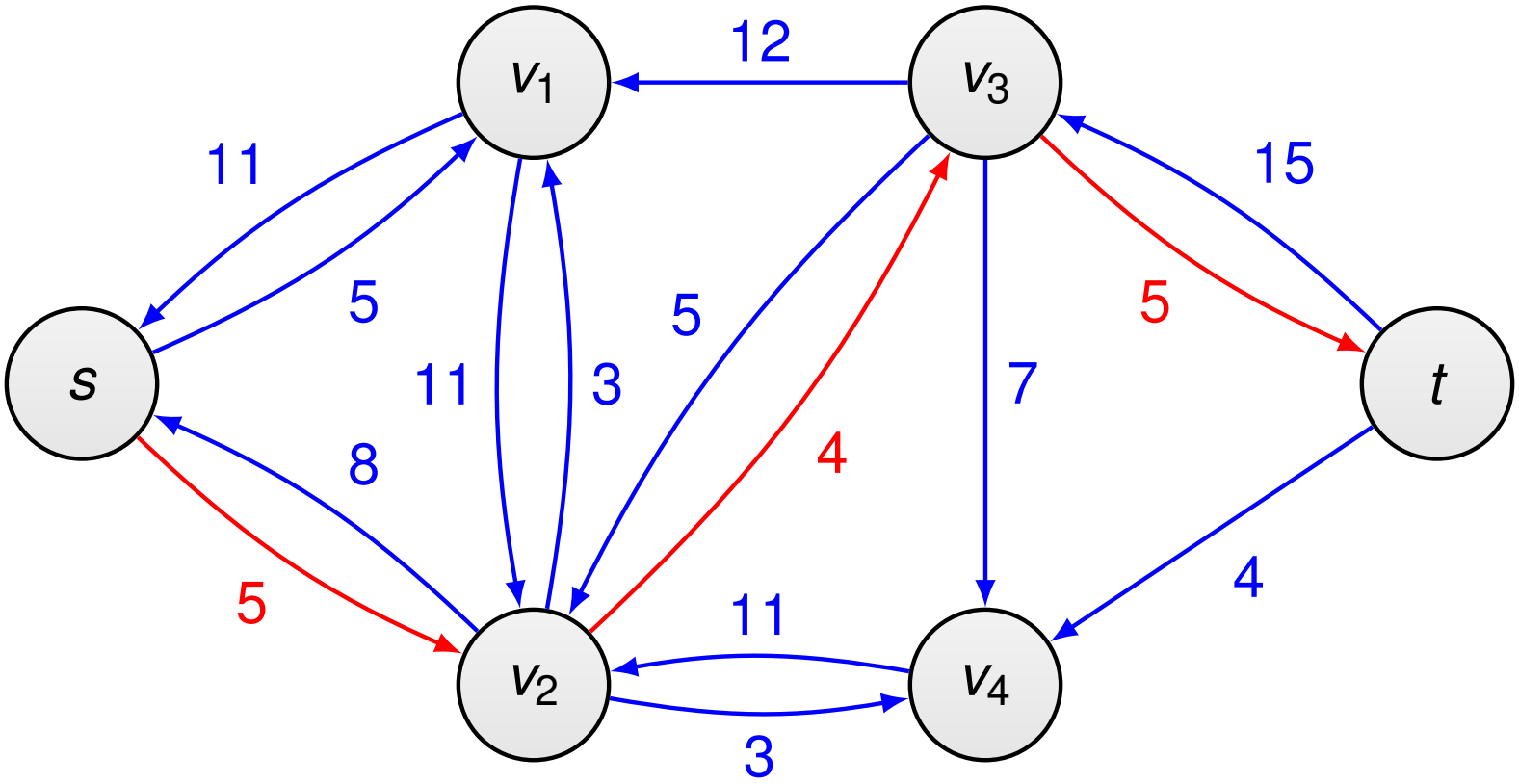
- Los vértices son los mismos que en la red original.
- Hay una arista de u a v con capacidad la suma de:
  - Lo que me falta para llenar la capacidad de u a v en la red original.
  - El flujo que estoy mandando de v a u.



Camino aumentante

Un camino aumentante es un camino de s a t en la red residual. Dado un camino aumentante:

- Podemos conseguir un flujo de mayor valor mandando todo lo posible por este camino, (capacidad del camino).



Método de Ford-Fulkerson

Esto induce el siguiente algoritmo de flujo máximo:

- Empiezo con el flujo vacío (todos 0).
- Construyo la red residual inicial para este flujo.
- Mientras haya camino de s a t en la red:
  - Busco el arco de menor peso (m) del camino.
  - Actualizo el flujo para mandar m unidades de flujo por ese camino.
  - Actualizo la red residual para el nuevo flujo.

  Devuelvo el flujo obtenido.
  La complejidad es O(hallar un s-t camino aumentante) $\times$ $O(F)$ donde F es el valor del flujo máximo.

Edmonds-Karp y Dinitz

Si además buscamos el camino de s a t con un BFS:

- El algoritmo queda $O(NM^2)$ con N la cantidad de vértices y m la de aristas.
- Además, siempre tenemos la cota $O(MF)$ donde F es el valor del flujo máximo.
- Este algoritmo se conoce como Edmond-Karp.

Otro algoritmo similar es el de Dinitz:

- Es $O(N^2M)$ por lo que es más rápido que E-K.
- Es más largo de implementar que E-K.

Hay algoritmos mejores pero en competencias con estos dos alcanza. Son fundamentales en cualquier notebook.

Aprovechar la red residual

Supongamos que nos piden saber si el flujo máximo en una red cambia si aumentamos en 1 la capacidad de una arista ¿Qué harían?

- Podemos calcular el flujo máximo para la red original.
- Luego calculamos el flujo máximo para la red nueva pero partiendo del flujo anterior.

Supongamos ahora que nos disminuyen la capacidad de una arista (por la que saturabamos la capacidad) en 1 ¿Qué harían?

- Buscamos un camino en la red residual de t a s que pase por la arista a disminuir.
- ”Retrotraemos” 1 de flujo por ese camino y disminuimos la capacidad.
- Calculamos el flujo máximo partiendo de este flujo.

En resumen, pensar siempre al correr varios flujos si no podemos reusar la información obtenida.

Aprovechar la red residual

Supongamos que nos piden saber si el flujo máximo en una red cambia si aumentamos en 1 la capacidad de una arista ¿Qué harían?

- Podemos calcular el flujo máximo para la red original.
- Luego calculamos el flujo máximo para la red nueva pero partiendo del flujo anterior.

Supongamos ahora que nos disminuyen la capacidad de una arista (por la que saturabamos la capacidad) en 1 ¿Qué harían?

- Buscamos un camino en la red residual de t a s que pase por la arista a disminuir.
- ”Retrotraemos” 1 de flujo por ese camino y disminuimos la capacidad.
- Calculamos el flujo máximo partiendo de este flujo.

En resumen, pensar siempre al correr varios flujos si no podemos reusar la información obtenida.

Aprovechar la red residual

Supongamos que nos piden saber si el flujo máximo en una red cambia si aumentamos en 1 la capacidad de una arista ¿Qué harían?

- Podemos calcular el flujo máximo para la red original.
- Luego calculamos el flujo máximo para la red nueva pero partiendo del flujo anterior.

Supongamos ahora que nos disminuyen la capacidad de una arista (por la que saturabamos la capacidad) en 1 ¿Qué harían?

- Buscamos un camino en la red residual de t a s que pase por la arista a disminuir.
- ”Retrotraemos” 1 de flujo por ese camino y disminuimos la capacidad.
- Calculamos el flujo máximo partiendo de este flujo.

En resumen, pensar siempre al correr varios flujos si no podemos reusar la información obtenida.

Max-Flow Min-Cut

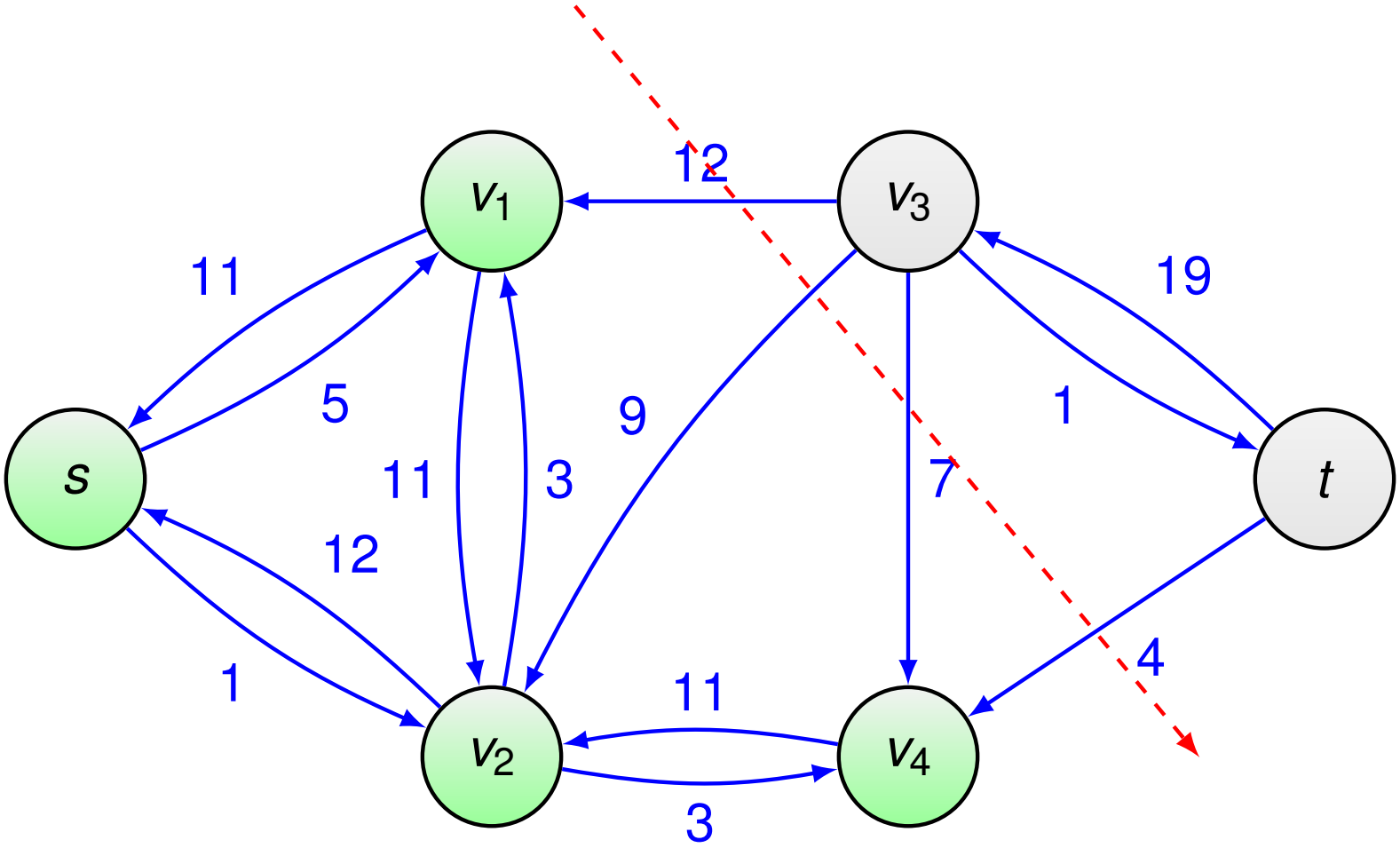
Un corte en una red es una partición de los vértices en dos conjuntos U y V tal que $s \in U$ y $t \in V$.
Dado un corte en una red su valor es:

- La suma de las capacidades de todos los arcos que van de un vértice de U a uno de V.

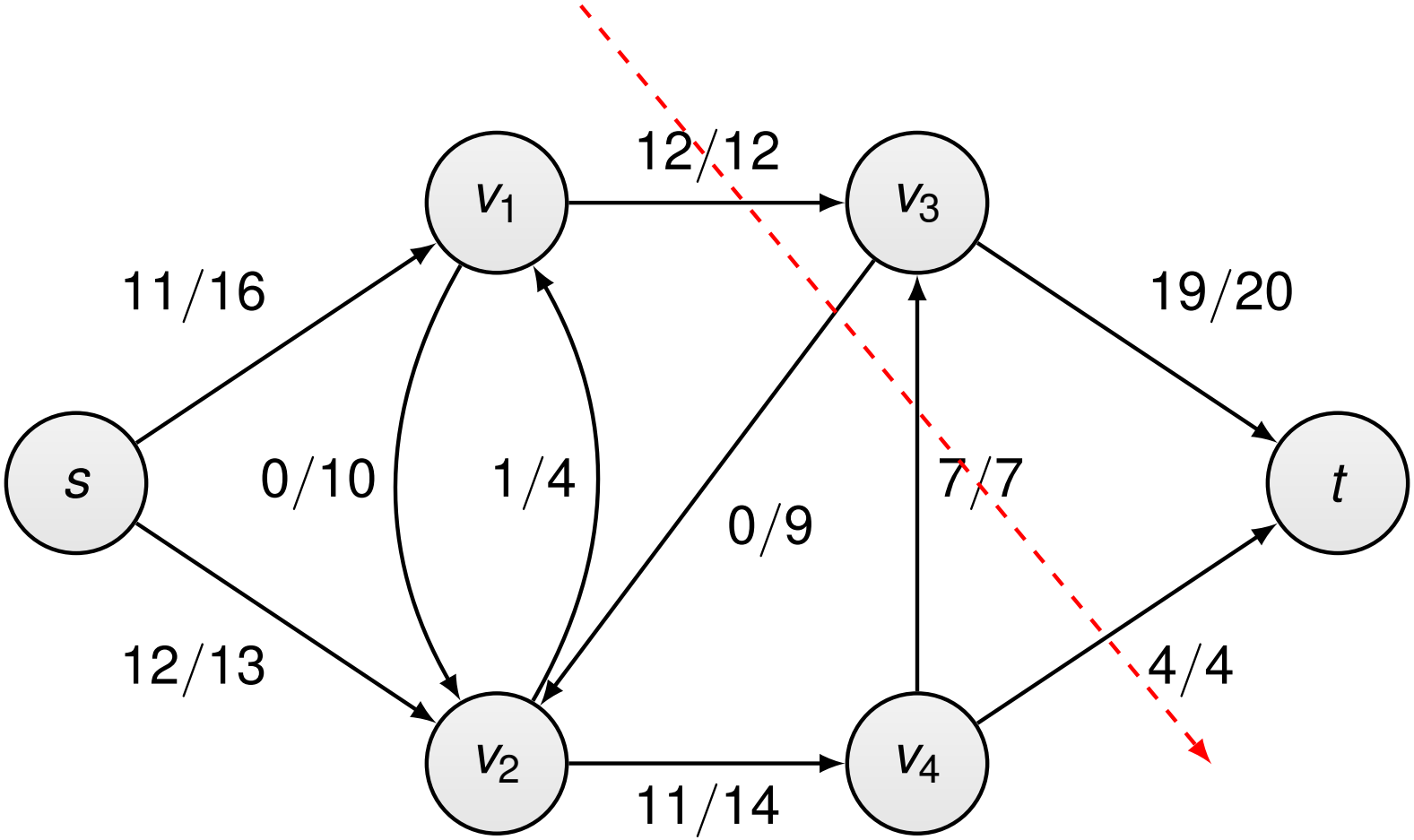
El teorema de Max-Flow Min-Cut nos dice que:

- El valor del corte mínimo coincide con el del flujo máximo.
- Podemos encontrar un corte mínimo definiendo:
  - U como los vértices alcanzables desde s en la red residual del flujo máximo.
  - V como el resto de los vértices.

Max-Flow Min-Cut



Max-Flow Min-Cut



Múltiples fuentes/sumideros

Supongamos que tenemos un problema donde hay varios vértices de donde ”sale” flujo y/o varios vértices a los que tiene que llegar flujo ¿Qué hacemos?

- Creamos dos vértices s y t nuevos.
- Conectamos s a todos los vértices de los que ”sale” flujo con capacidad ∞
- Conectamos todos los vértices a los que llega flujo con t con capacidad ∞.

Múltiples fuentes/sumideros

Supongamos que tenemos un problema donde hay varios vértices de donde ”sale” flujo y/o varios vértices a los que tiene que llegar flujo ¿Qué hacemos?

- Creamos dos vértices s y t nuevos.
- Conectamos s a todos los vértices de los que ”sale” flujo con capacidad ∞
- Conectamos todos los vértices a los que llega flujo con t con capacidad ∞.

Restricciones sobre los vértices

¿Qué pasa si tenemos restricciones de flujo que pasa por un vértice v en lugar de por un arco?

- Duplicamos el vértice en dos vértices v_{in} y v_{out}.
- Conectamos todas las aristas que entraban a v a v_{in}.
- Conectamos todas las aristas que salían de v a v_{out}.
- Conectamos v_{in} a v_{out} con un arco de la capacidad máxima del vértice.

Restricciones sobre los vértices

¿Qué pasa si tenemos restricciones de flujo que pasa por un vértice v en lugar de por un arco?

- Duplicamos el vértice en dos vértices v_{in} y v_{out}.
- Conectamos todas las aristas que entraban a v a v_{in}.
- Conectamos todas las aristas que salían de v a v_{out}.
- Conectamos v_{in} a v_{out} con un arco de la capacidad máxima del vértice.

Matching máximo

Dado un grafo, un matching es un conjunto de aristas tal que no haya dos que incidan sobre el mismo vértice.
El problema de matching máximo pide el mayor conjunto de aristas que cumpla esto.
Si el grafo es bipartito se puede modelar con un flujo de la siguiente manera:

- Si P y Q son los conjuntos de vértices de cada lado.
- Creo un vértice s y lo conecto con capacidad 1 a cada vértice de P.
- Creo un vértice t y conecto cada vértice de Q a t con capacidad 1.
- Por cada arista (p, q) del grafo uno a p con q con capacidad ∞.

El flujo máximo sobre esta red nos da el matching máximo Las aristas del medio por las que enviamos flujo forman el matching.

Mínimo cubrimiento por aristas

Dado un grafo, un cubrimiento por aristas es un conjunto de aristas tal que todo vértice sea extremo de alguna arista del conjunto. Ahora nos interesa el cubrimiento más chico.
Si el grafo es bipartito se puede obtener a partir del matching máximo:

- Agrego al cubrimiento todas las aristas de un matching máximo.
- Por cada vértice que me quedó sin cubrir agrego una arista cualquiera que incida en el vértice.

De esta manera, el tamaño del cubrimiento mínimo es $N - \mathcal{M}$ donde $\mathcal{M}$ es el tamaño del matching máximo.

Mínimo cubrimiento por vértices

Dado un grafo, un cubrimiento por vértices es un conjunto de vértices tal que toda arista tenga algún extremo en el conjunto. Nuevamente nos interesa el cubrimiento más chico.
Si el grafo es bipartito:

- El teorema de König nos dice que el tamaño del cubrimiento mínimo coincide con el del matching máximo.

Para obtener el cubrimiento:

- Hallar el flujo máximo sobre la red del problema de matching máximo.
- Busco un corte mínimo.
- Un cubrimiento mínimo son los extremos de las aristas cortadas que no son s ni t.

Máximo conjunto independiente

Dado un grafo, un conjunto independiente es un conjunto de vértices tal que no hay dos adyacentes. Queremos encontrar el conjunto independiente más grande.
Si el grafo es bipartito:

- Tomo un cubrimiento por vértices mínimo.
- Los vértices que no están en el cubrimiento forman un conjunto independiente máximo.

Nuevamente, el tamaño del conjunto independiente más grande es $N - \mathcal{M}$.

Sobre la complejidad

A la hora de calcular un matching máximo con flujo.
Si uso E-K:

- El flujo máximo es a lo sumo $O(N)$.
- Usando la cota $O(MF)$ nos queda que matching máximo es $O(NM)$.

Si uso Dinitz:

- Se puede probar que en la red de matching máximo tarda $O(M\sqrt{N})$.

Problemas sobre tableros

Es muy común tener alguno de estos problemas escondido en algún problema de tableros.
Conviene pensar el siguiente modelo:

- Creo un grafo bipartito.
- Creo un vértice en P por cada fila del tablero.
- Creo un vértice en Q por cada columna del tablero.
- Una arista entre un vértice fila y un vértice columna representa la casilla de esa fila y columna.

Muchas veces en este modelo el problema se reduce a alguno de los 4 que vimos.
Ejemplo: Costume Change

Flujo máximo de costo mínimo

Una variante del problema de flujo máximo es la siguiente:

- A cada arco, le agregamos un costo de mandar una unidad de flujo por ese arco.
- El costo de un flujo es la suma de los costos de las aristas multiplicado por el flujo que mandamos por cada una.
- Queremos encontrar entre todos los flujos máximos, aquel que minimice el costo total.

Se puede resolver con ideas similares:

- Si corro el algoritmo usual pero busco un camino aumentante con Bellman-Ford en los costos obtengo uno de costo mínimo. Complejidad $O(NMF)$.
- Se puede hacer una modificación para que los costos no sean negativos y correr Dijkstra. Complejidad $O(MF \log N)$.

Matching máximo de costo mínimo

La versión de matching máximo de costo mínimo es similar y se puede resolver así:

- Usando el Hungarian Algorithm que lo resuelve en $O(n^3)$.
- Modelándolo igual que antes pero con costos en las aristas y usar el algoritmo de Max Flow Min Cost.

Ejemplo: Direction Board

Dilworth

Dado un grafo acíclico dirigido (DAG) tenemos las siguientes definiciones:

- Una cadena es una secuencia de vértices tal que de cada uno puedo llegar al siguiente.
- Una anticadena es un conjunto de vértices tal que no hay camino entre ningún par de ellos.

El teorema de Dilworth nos dice que los tamaños de estas cosas coinciden:

- La anticadena de mayor tamaño
- La menor partición en cadenas de los vértices del grafo.

Ejemplo: Stock Charts

Más problemas:

- Inspection: goo.gl/iYho0F
- Super Watch: goo.gl/Y1h5zY
- Oil Company: goo.gl/XW3jSp
- Shogui Tournament: goo.gl/KRz3Yi
- Graduation: goo.gl/hnCoOU
- Parking: goo.gl/21ZJqC
- "Shortest"pair of paths: goo.gl/ardYaY
- Angels and Devils: goo.gl/sCkd30

Los algoritmos en detalle (inglés):

- Max flow: goo.gl/luDODH
- Min-cost max-Flow: goo.gl/XfRWjS
- Hungarian: https://tinyurl.com/ycto9f3n