1 Lezione del 19-11-24

1.0.1 Algoritmo di Ford-Fulkerson

L'algoritmo di **Ford-Fulkerson** (precisamente nella variante di *Edmonds-Karp*) è un algoritmo per il calcolo del flusso massimo su reti.

Iniziamo con alcune definizioni, prima fra tutte quella di **taglio di rete**:

Definizione 1.1: Taglio di rete

Dato un problema di flusso massimo, un taglio di rete è una partizione dei nodi in due sottoinsiemi N_s e N_t, quindi con $N_s \cup N_t = N$ e $N_s \cap N_t = \emptyset$, con il vincolo aggiunto che la sorgente di flusso s deve trovarsi $\in N_s$, e la destinazione del flusso t deve trovarsi $\in N_t$.

Possiamo quindi dare la definizione di **arco diretto del taglio**:

Definizione 1.2: Arco diretto del taglio

Dato un taglio, si dicono archi diretti del taglio tutti gli archi (i, j) che hanno $i \in N_s$ e $j \in N_t$.

con enfasi sul fatto che un arco diretto del taglio va da N_s a N_t, e **non** viceversa.

Infine, su questi archi diretti del taglio definiamo la **portata**:

Definizione 1.3: Portata del taglio

Dato un taglio (N_s, N_t), la portata $\mu(N_s, N_t)$ viene definita come:

$$\mu(N_s, N_t) = \sum_{(i,j) \in A^T} u_{ij}$$

Notiamo come ogni taglio di rete rappresenta un limite superiore sul flusso massimo:

$$\overline{x} \leq \mu(N_s, N_t)$$

Possiamo quindi dimostrare il teorema:

Teorema 1.1: Massimo flussimo - minimo taglio

Il flusso massimo di un problema di flusso massimo con soluzione $\neq -\infty$ è uguale al taglio di portata minima:

$$\overline{v} = \overline{u}(N_s, N_t) = \min_{s,t} \mu(N_s, N_t)$$

Notiamo come le ultime due equazioni ricalcano rispettivamente la **dualità debole** e la **dualità forte**: si può effettivamente dimostrare che il problema del taglio minimo è il **duale** del problema di flusso massimo (effettivamente vogliamo trovare un *limite superiore* del flusso, che è lo stesso procedimento che avevamo adottato,, in quel caso adattando le diseguaglianze, nel ricavare il duale nei problemi di LP).

Chiaramente è improbabile calcolare il taglio minimo per enumerazione completa: togliendo il fatto che le partizioni dei tagli lavorano su $n - 2$ anziché n nodi (in quanto

ci sono i vincoli $s \in N_s$ e $t \in N_t$), il numero di partizioni va comunque come $\sim 2^n$, che è quindi a complessità **esponenziale**.

Vediamo allora quale algoritmo possiamo usare per ricavare i tagli minimi (e quindi i flussi massimi) in maniera efficiente.

1.0.2 Definizione dell'algoritmo

Prendiamo in considerazione un grafo e il suo **grafo residuo**, cioè il grafo che complementa tutti gli archi (diretti) con un arco (diretto) identico ma di verso opposto. Chiameremo *archi reali* gli archi che fanno effettivamente parte del grafo, e *archi fittizi* gli archi che introduciamo sul grafo residuo.

Per ogni arco (i, j), reale o fittizio, definiamo la **capacità residua** come:

- Archi reali: $r_{ij} = u_{ij} - \overline{x}_{ij}$
- Archi fittizi: $r_{ji} = \overline{x}_{ij}$

assunto nelle definizioni che (i, j) sia l'arco reale e di conseguenza (j, i) l'arco fittizio.

A parole, la capacità residua dell'arco *reale* è *quanto posso ancora spedire sull'arco*, mentre la capacità residua dell'arco *fittizio* è *quanto ho già spedito sull'arco vero*, da cui:

$$r_{ij} + r_{ji} = u_{ij} - \overline{x}_{ij} + \overline{x}_{ij} = u_{ij}$$

cioè la somma delle capacità residue su entrambi gli archi nelle due direzioni del grafo residuo dà sempre la capacità massima sull'arco reale nella stessa posizione del grafo originale.

Definiamo allora il concetto di **cammino aumentante** sul grafo:

Definizione 1.4: Cammino aumentante

Dato un problema di flusso massimo, chiamiamo cammino aumentante C_{aum} sul grafo residuo un qualsiasi cammino orientato da s a t formato da archi con capacità residue $r_{ij} > 0$.

Di un dato cammino aumentante C_{aum}, il dato interessante è la **portata** di C_{aum}:

$$\delta = \min_{i,j \in C_{aum}} \{r_{ij}\}$$

Il primo passo dell'algoritmo di Ford-Fulkerson sarà allora selezionare un cammino aumentante e calcolarne la portata δ. In seguito vorremo spedire queste δ unità di flusso lungo il cammino, quindi applicando la regola:

$$\overline{x}' = \begin{cases} \overline{x}_{ij} + \delta, & \forall (i,j) \in C_{aum} \\ \overline{x}_{ij}, & \forall (i,j) \notin C_{aum} \end{cases}$$

A questo punto basta ricalcolare le varie capacità residue r_{ij} e ripetere il passaggio. Chiaramente, quando non si riuscirà più a trovare cammini aumentanti, cioè archi con capacità residue > 0, significherà che avremo trovato la soluzione ottima.

Algoritmo 1 di Ford-Fulkerson

Input: un problema di flusso massimo
Output: la soluzione ottima
ciclo:
Stabilisci un cammino aumentante C_{aum} e calcola la portata δ.
if $C_{aum} = \emptyset$ **then**
Fermati, sei all'ottimo
end if
Aggiorna il flusso $\overline{x}$ secondo la regola:

$$\overline{x}' = \begin{cases} \overline{x}_{ij} + \delta, & \forall (i,j) \in C_{aum} \\ \overline{x}_{ij}, & \forall (i,j) \notin C_{aum} \end{cases}$$

Ricalcola le capacità residue r_{ij}
Torna a ciclo

Si presenta un algoritmo per il calcolo dei cammini aumentanti, detto **algoritmo della croce**:

Algoritmo 2 della croce

Input: il grafo residuo di un problema di flusso massimo
Output: un cammino aumentante C_{aum}
Inizializza due vettori, closed e open
Inserisci s, il nodo di partenza, in closed
Inserisci tutti i nodi raggiungibili da closed, cioè quelli che sono collegati con un arco diretto dal nodo 1 con costo residuo > 0, in open
ciclo:
Prendi il primo elemento di open e mettilo in closed
Inserisci tutti i nodi raggiungibili da closed, cioè quelli che sono collegati con un arco diretto dal nodo in closed con costo residuo > 0
if non hai raggiunto t; cioè il nodo di arrivo **then**
Torna a ciclo
end if
Ripercorrendo i nodi in closed al contrario, partendo da quello di arrivo, hai il cammino aumentate C_{aum}

Notiamo come, nell'applicazione dell'algoritmo della croce, è necessaria la presenza degli archi residui in entrambe le direzioni: potrebbe infatti succedere che un arco si svuoti nella direzione canonica e si riempa nella direzione opposta. Questo significherà che il flusso che spingiamo nella direzione opposta andrà a sottrarsi al flusso già presente sull'arco (in altre parole, l'algoritmo di Ford-Fulkerson ammette di "cambiare idea" nell'esplorazione dello spazio delle soluzioni).