1 Lezione del 11-11-24

1.1 Problemi esprimibili come flussi minimi

Diversi problemi che abbiamo già visto possono essere formulati come problemi di flusso minimo su grafi. In particolare, vediamo l'**assegnamento di costo minimo** e il problema di **trasporto**.

1.1.1 Trasporto

Avevamo definito un problema di trasporto come un problema LP in forma:

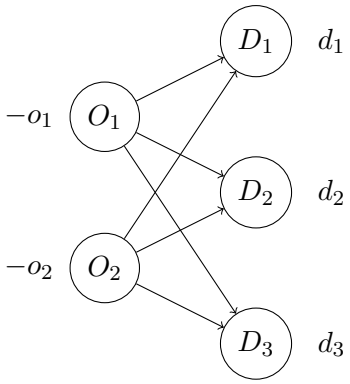
$$\begin{cases} \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{i=1}^{m} x_{ij} \geq d_j, \quad \forall i = 1, ..., n \\ \sum_{j=1}^{n} x_{ij} \leq o_{ij}, \quad \forall j = 1, ..., m \\ x_{ij} \geq 0 \end{cases}$$

Possiamo concettualizzare un problema di questo tipo come un problema di flusso minimo su un **grafo bipartito**:

Definizione 1.1: Grafo bipartito

Si dice **bipartito** un grafo dove i nodi N possono essere divisi in due insiemi disgiunti U e V, dove ogni arco collega un nodo U a un nodo V.

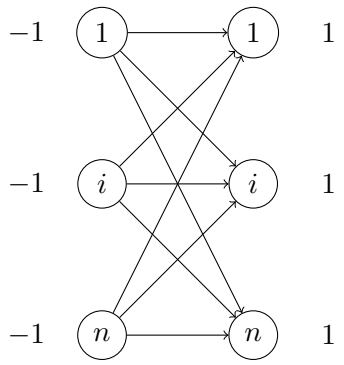
Graficamente, il grafo bipartito di un problema di trasporto ha la forma:



dove ogni nodo di **origine** O_i eroga o_i unità di flusso, e ogni nodo di domanda D_i richiede d_i unità di flusso.

1.1.2 Assegnamento di costo minimo

L'assegnamento di costo minimo può intedersi come un trasporto a **volume unitario**, cioè ogni nodo di origine eroga una singola unità di flusso, e ogni nodo di domanda richiede una singola unità di flusso. Chiaramente, per n nodi di domanda dovremo avere n nodi di origine (cioè, per ogni "spazio" assegnabile abbiamo bisogno di un possibile assegnamento). Questo si traduce sempre in un grafo bipartito, nella forma:



1.2 Cammini minimi

Abbiamo visto come l'ottimizzazione sui grafi può essere usata per risolvere problemi di flusso di costo minimo. Vediamo un'altro tipo di problema che può essere risolto attraverso una versione modificata del simplesso sui grafi: quello dei **cammini a costo minimo**.

In generale, un problema dei cammini a costo minimo è un problema di ottimizzazione sui grafi in forma:

$$\begin{cases} \min c^T \cdot x \\ Ex = b \\ x \geq 0 \end{cases}$$

che dà i cammini minimi da un nodo r a tutti gli altri nodi $i \neq r$. Questo si ottiene scegliendo il vettore b appositamente:

$$b_i = \begin{cases} -(n-1), & i = r \\ 1, & i \neq r \end{cases}$$

cioè impostando il nodo r come una sorgente da $n-1$ unità, e ogni nodo di arrivo $i \neq r$ come un pozzo di 1 unità.

Notiamo che un albero di copertura T_r radicato in r rappresenta una base ammissibile del problema, in quanto porterà le $-(n-1)$ unità di flusso sugli $n-1$ nodi. Inoltre, il flusso di base asssociato a T_r sarà **non degenere**, cioè nessuno flusso sarà uguale a 0 (ogni nodo avrà in arrivo almeno 1 unità di flusso). Possiamo quindi dire che il teorema di Bellman da solo basta a dimostrare l'ottimalià, e non occorrono regole anticiclo di Bland (non esistono soluzioni di base ammissibili degeneri). Anzi, possiamo dire che conviene prendere l'arco con costo ridotto minimo, in quanto comporterà l'abbattimento maggiore del valore ottimo.

Infine, possiamo fare un'ottimizzazione significativa per quanto riguarda la rimozione degli archi in $\mathcal{C}^-$ (cioè quelli discordi all'arco introdotto (p, q)) nella fase di **rimozione dei cicli**. Si ha che a ogni passso dell'algoritmo la soluzione ammissibile considerata è un albero di copertura, ergo ogni nodo sarà raggiunto. L'algoritmo di Bellman restituirà quindi un modo *più efficiente* di quello previsto dall'albero di arrivare ad un dato nodo q, partendo da un nodo p. Avremo quindi che esiste sempre un nodo (i, q) **entrante** in q, cioè il modo di arrivare in q che era originariamente previsto dall'albero che abbiamo appena considerato. Inoltre, si avrà che questo arco sarà l'ultimo a portare a q, o almeno sarà a costo minore dei suoi precedenti (ogni precedente dovrà portare a q e a tutti gli eventuali nodi a cui arriva q). Ergo, (i, q) è a **flusso minimo**. Possiamo quindi semplificare la regola dell'arco uscente per la rimozione di cicli.

Formuliamo allora il **simplesso per cammini**:

Algoritmo 1 del simplesso per cammini

Input: un problema di cammini di costo minimo
Output: la soluzione ottima
Trova un albero T di radice r
ciclo:
Calcola il potenziale di base $\bar{\pi}^T = c_T^T E_T^{-1}$
Calcola i costi ridotti $c_{ij}^{\bar{\pi}} = c_{ij} + \bar{\pi}_{ij} - \bar{\pi}_{ij}$ per ogni arco
if $c_{ij}^{\overline{\pi}} \geq 0 \ \ \forall (i,j) \in L$ **then**
Fermati, $\bar{x}$ è un albero dei cammini minimi
else
Calcola l'arco entrante:

$$(p,q) = (i,j) \in L : c_{ij}^{\bar{\pi}} = \min\{c_{ij}^{\bar{\pi}}\}$$

Chiama $\mathcal{C}$ il ciclo che l'arco (p,q) forma con gli archi in T
Fissa un orientamento concorde a (p,q) su $\mathcal{C}$ e partiziona $\mathcal{C}$ in $\mathcal{C}^+$ archi concordi e $\mathcal{C}^-$ archi discordi a tale orientamento
end if
if $\mathcal{C}^- = \emptyset$ **then**
Fermati, non esiste un albero dei cammini minimi
else
Scegli come arco uscente l'unico arco $(i,q) \in T$ che entra in q
end if
Aggiorna l'albero come:

$$T = T \setminus \{(r,s)\} \cup \{(p,q)\}$$

Torna a ciclo

1.2.1 Cammini minimi multiobiettivo

Problema 1.1: Cammini minimi a due obiettivi

Vogliamo progettare un'applicazione di navigazione per dispositivi cellulari. L'applicazione usa una struttura dati a grafo per rappresentare località e le strade che collegano. Per ogni strada, si tiene conto di una distanza d_{ij} in KM, e di un costo c_{ij} al pedaggio. Si vuole trovare un algoritmo che trovi il percorso ottimo fra due località i e j.

Il problema presentato è un di ottimizzazione **multiobiettivo**: vogliamo *minimizzare* sia il costo che la distanza. Un'approccio è quello di impostare il problema di costo minimo sulle distanze, cioè come:

$$\begin{cases} \min d^T \cdot x \\ Ex = b \\ x \geq 0 \end{cases}$$

e di introdurre termine di *budget*, cioè un valore massimo C di costo che siamo disposti a pagare:

$$c^T x \leq C$$

Questo approccio però non è propriamente ottimale, in quanto introducendo un nuovo vincolo, che potrebbe violare le condizoni di **interezza** rispettate dalla matrice di adiacenza, che sappiamo essere **unimodulare**. Per continuare a prendere archi interi, dovremmo obbligatoriamente introdurre un vincolo $x_{ij} \in \{0, 1\}$, ergo trasformare il problema in un problema di ILP. Vedremo quindi metodi più efficienti per ottimizzare problemi multiobiettivo di questo tipo.