Algoritmi semplici (per davvero)

Esercizio → metric matching on the line

Sia $S = s_1, s_2, \dots, s_n$ un insieme di punti ordinati sulla retta reale, rappresentanti dei server.
Sia $C = c_1, c_2, \dots, c_n$ un insieme di punti ordinati sulla retta reale, rappresentanti dei client.
Il costo di assegnare un client c_i ad un server s_j è $|c_i - s_j|$. Si fornisca un algoritmo greedy che assegna ogni client ad un server distinto e che minimizzi il costo totale dell'assegnamento.

Spiegazione <u>completa</u> del problema (mai data esplicitamente dal prof, tradotta da ricerche sue)

SI riduce ad un problema di assegnazione dei vertici su un grafo, prendendo come limite massimo dell'assegnazione $2n - 1$ vertici e cercando per ognuno la sequenza più corta assegnata.
Dovendo assegnare ciascun client a ciascun server, partiamo con l'attuare una scelta greedy → scegliamo l'ultimo client disponibile in ordine di tempo e gli assegniamo il server più vicino, l'ultimo presente.
In questo modo, il costo della richiesta, che comunque può essere randomico, tende a minimizzarsi.
Matematicamente, per una richiesta con costo c_i^*avremo una probabilità $1/n$ di beccare la scelta corretta e assumiamo che $\forall j \in n$

$$c_j \le c_j^* + \frac{1}{n-j+1}\sum_{i=1}^{j-1} c_i + \frac{1}{n-j+1}\sum_{i=1}^{j-1} c_i^*$$

Per la prova della dimostrazione, sia $j \in [n]$ arbitrario e fissiamo l'assegnazione delle richieste ai server fino alla richiesta $r_{(j-1)}$. Per $i \in [j-1]$, sia $s(r_i)$ denota il server assegnato alla richiesta r . Per di più, per $i \in [n]$ sia $s^*(r_i)$ il server assegnato alla richiesta r_i in un'assegnazione ottimale fissa OPT. Consideriamo il grafo ausiliario diretto G in cui l'insieme dei vertici corrisponde alle richieste, ed esiste un arco diretto (r_i, r_j) se $s(r_i) = s^*_{(rj)}$, i.e., la richiesta r_i è associata al server assegnato a r_j in OPT. (Si ammettono anche gli auto-loop quando le richieste vengono abbinate allo stesso server nella soluzione ottima e in quella greedy). Quando la richiesta r_j arriva, G ha esattamente $j - 1$ archi, ogni vertice è il punto finale di un solo arco e non c'è nessuno arco in uscita dal vertice corrispondente alla richiesta r_j . Condizionato dal fatto che r_j non è il punto finale di un arco, abbiamo $c_j \le c_j^*$ come $s^*_{(rj)}$ non è stato assegnato a una richiesta r_i , $i \in 1, \dots, j-1$, ae, quindi, il costo dell'abbinamento r_j è limitato dal costo in OPT. Se, d'altro canto, r_j è il punto finale di un arco, sia $P = (i_1, \dots, i_l = r_j)$ con $l \le j-1$ sia il percorso diretto massimo in G che termina in r_j . Allora, per la massimizzazione di P, il server $s^*(r_{i_1})$ è disponibile e, quindi, il costo atteso della j-esima richiesta può essere limitato dall'alto (per la disuguaglianza triangolare) da

$$c_j \le c_j^* + \sum_{k=1}^{\ell}(c_{i_k} + c_{i_k}^*),$$

Infine, si noti che per una richiesta $r_i, i \in [j-1]$, il termine $(c_i + c_i^*)$ appare nel limite superiore c_j nella immagine precedente se e solo se la richiesta r_j è il punto finale del percorso unico che utilizza il bordo che ha origine in r_i . In altre parole, tra le rimanenti $n-j+1$ richieste, ce ne sta esattamente una per cui il termine $(c_i + c_i^*)$ appare nel calcolo di c. Per ciascuna delle rimanente $n - j + 1$ richieste, una è selezionata con probabilità $1/(n-j+1)$, e otteniamo

$$c_j \le c_j^* + \frac{1}{n-j+1}\sum_{i=1}^{j-1}(c_i + c_i^*),$$

Per concludere, avremo quindi che:

$$\sum_{i=1}^{j} c_i \le c_j^* + \left(1 + \frac{1}{n-j+1}\right)\sum_{i=1}^{j-1} c_i + \frac{1}{n-j+1}\sum_{i=1}^{j-1} c_i^*$$

Scritto da Gabriel

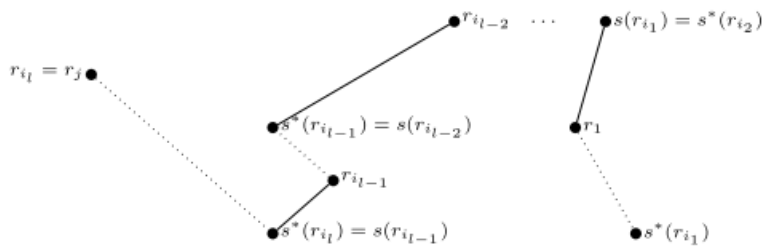

Figura 1. Server disponibili per la richiesta $r_j$nella dimostrazione del Teorema 1. Le linee solide corrispondono ai costi che si verificano nella soluzione greedy, mentre le linee tratteggiate corrispondono ai costi che si verificano nella soluzione ottimale. Poiché i costi sono metrici, il costo $d\left(r_j, s^*(r_{i_1})\right)$ è limitato dalla lunghezza del percorso

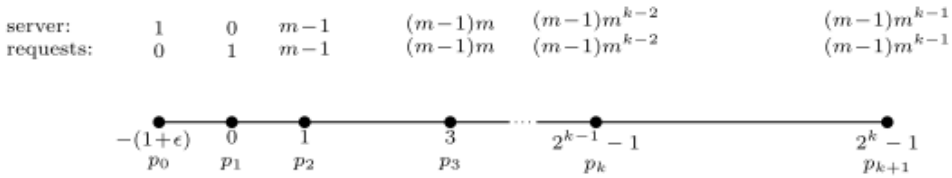

Fig. 2. Lower bound.

Quindi il costo ottimale è dato, usando $c_i^* = OPT/n$, otteniamo:

$$\sum_{i=1}^{j} c_i \le \frac{n-j+2}{n-j+1}\sum_{i=1}^{j-1} c_i + \left(\frac{j-1}{n-j+1}+1\right)\frac{\mathrm{OPT}}{n}$$
$$= \frac{n-j+2}{n-j+1}\sum_{i=1}^{j-1} c_i + \frac{\mathrm{OPT}}{n-j+1}.$$

e ricorsivamente:

$$\sum_{i=1}^{j} c_i \le \frac{n}{n-j+1}c_1 + \frac{j-1}{n-j+1}\mathrm{OPT}$$
$$= \frac{1}{n-j+1}\mathrm{OPT} + \frac{j-1}{n-j+1}\mathrm{OPT}$$
$$= \frac{j}{n-j+1}\mathrm{OPT}.$$

Questo dimostra che per $j = n$ otteniamo il risultato sperato (che tradotto, conferma che partendo dalla fine del vettore server ed assegnano sulla base dei client, si riesce come dimostrato matematicamente a massimizzare l'assegnazione).

Alla luce di questo ambaradan, l'algoritmo greedy è:

$METRIC - MATCHING\ (S, C)$
$n = C.length$
$A = C[n]$ $\quad // prende\ l'ultimo\ elemento$
$last \Rightarrow 1$
$for\ j = 2\ to\ n$
$\quad for\ j = 1\ to\ n-1$ $//mi\ assicura\ di\ prendere\ anche\ la\ richiesta\ precedente$
$\quad\quad if\ \ C_i - S_j > C_{(last)} - S_{(last)} + 1\ then$
$\quad\quad\quad last \Leftarrow j$ $\quad // mi\ assicuro\ di\ salvare\ il\ server\ che\ soddisfa\ la\ richiesta$
$\quad\quad\quad A \Leftarrow [C_i, C_i + 1] \cup A$
$return\ A$