Algoritmi greedy
VII parte

Progettazione di Algoritmi a.a. 2023-24
Matricole congrue a 1
Docente: Annalisa De Bonis

146

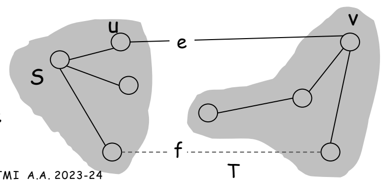
Proprietà del ciclo

- Proprietà del ciclo . Sia C un ciclo e sia e=(u,v) un arco di costo massimo tra quelli appartenenti a C. Esiste un minimo albero ricoprente che non contiene l'arco e.

Dim. (tecnica dello scambio)

- Sia T un albero ricoprente che contiene l'arco e. Dimostriamo che possiamo sostituire e con un altro arco di C in modo da ottenere ancora uno MST.
- Se rimuoviamo l 'arco e da T disconnettiamo T in due alberi uno contenente u e l'altro contenente v. Chiamiamo S l'insieme dei nodi dell'albero che contiene u.
- Il ciclo C contiene due percorsi per andare da u a v. Un percorso è costituito dall'arco e=(u,v) mentre l'altro va da u a v attraverso gli archi di C diversi da (u,v). Tra questi archi deve essercene uno che attraversa il taglio [S,V-S] altrimenti non sarebbe possibile andare da u che sta in S a v che sta in V-S. Sia f questo arco.

Se al posto di e inseriamo in T l'arco f, otteniamo un albero ricoprente T' di costo $c(T')=c(T)- c_e+c_f$
Siccome $c_f \leq c_e$ allora $c(T') \leq c(T)$. Siccome T è per ipotesi uno MST allora deve essere c(T') = c(T) → T' è anch'esso un MST.



147

Proprietà del ciclo

- Notiamo che se nella dimostrazione della proprieta` del ciclo l'arco f che attraversa il taglio (S,V-S) ha costo strettamente minore di e allora quando sostituiamo e con f otteniamo c(T') < c(T).
- Si ha quindi una contraddizione al fatto che T' sia un MST e che contenga e → non è possibile che uno MST contenga e.
- In altre parole se e è l'unico arco di costo massimo su un certo ciclo C allora nessun MST contiene e.
- Se invece l'arco f di C che attraversa il taglio (S,V-S) ha anch'esso costo massimo allora e puo` essere sostituito da f e l'albero ricoprente risultante avra` lo stesso costo di T.

Correttezza dell'algoritmo Inverti-Cancella

L'algoritmo Inverti-Cancella produce un MST.

Dim. Sia T il grafo prodotto da Inverti-Cancella.

Sia R_i l'insieme di archi rimossi fino ad un certo passo i da Inverti-Cancella e sia G_j il grafo ottenuto rimuovendo gli archi di R_i da G.

Dimostriamo per induzione che per ogni i esiste un MST che non contiene nessuno degli archi in R_i. La **base per i=0** e` banalmente verificata visto che non è stato cancellato ancora niente.

Passo induttivo. Supponiamo che per i<=j esista un MST di G che non contiene nessuno degli archi di R_i e dimostriamo che cio` vale anche $R_{j+1}$. Si noti che dire che esiste un MST di G che non contiene archi di $R_i$ equivale a dire che un MST di $G_i$ e` un MST di G.

Se al passo j+1 l'algoritmo non cancella l'arco e_{j+1} esaminato allora il passo induttivo e dimostrato banalmente perche' R_{j+1}=R_j.

Se al passo j+1 l'algoritno cancella l'arco e_{j+1} allora vuol dire che l'arco e_{j+1} si trova su un ciclo C **del grafo** G_j (altrimenti la sua rimozione avrebbe disconnesso le estremita di e_{j+1}).

continua

Correttezza dell'algoritmo Inverti-Cancella

- Dal momento che gli archi vengono esaminati in ordine non crescente di costo, l'arco e_{j+1} ha costo massimo tra gli archi sul ciclo C di G_j
  - La proprietà del ciclo implica allora che esiste un MST T_i di G_j che non contiene e_{j+1}.
  - Per ipotesi induttiva T_j e` anche un MST di G.
  - Siccome G_j non contiene alcun arco di R_j allora T_j non contiene ne' e_{j+1} ne' alcun arco di R_j
  - Abbiamo dimostrato che esiste un MST di G che non contiene alcun arco di $R_j \cup \{e_{j+1}\}= R_{j+1}$

150

150

Esercizio 13 Cap. 4

- Una piccola ditta che si occupa di fotocopiare documenti ogni giorno riceve le richieste di n clienti. La ditta dispone di un'unica fotocopiatrice e fotocopiare i documenti dell'i-esimo cliente richiede tempo t_i. A ciascun cliente i è associato un peso w_i che indica l'importanza del cliente i.
- Indichiamo con C_i il tempo in cui viene terminata la copia dei documenti del cliente i. Se i documenti del cliente i vengono fotocopiati per primi allora $C_i=t_i$. Se i documenti di i vengono fotocopiati dopo quelli di j allora $C_i=C_j+t_i$
- Vogliamo eseguire le fotocopie in un ordine che minimizzi $\sum_{i=1}^{n} w_i C_i$

Soluzione.

- Strategia greedy: Ordiniamo le richieste in modo non crescente rispetto ai valori w_i/t_i
- Dimostriamo che la soluzione greedy è ottima utilizzando la tecnica dello scambio.
- Sia G l'ordinamento greedy e O un ordinamento ottimo. Supponiamo $G \neq O$
- Esistono due richieste j e k tali che j precede k in G e k precede j in O. Inoltre devono esistere due richieste siffatte disposte una dopo l'altra in G .
  - Consideriamo le richieste k e j piu` vicine in $G$ per cui risulta k precede j in $G$ e j precede k in $O$. Se esistesse una richiesta i che in $G$ viene eseguita tra k e j questa dovrebbe essere eseguita dopo k anche in $O$ altrimenti k e i sarebbero due richieste eseguite in ordine diverso in $G$ e sarebbero tra di loro piu` vicine di k e j . Per la stessa ragione i dovrebbe essere eseguita prima di j in O. Cio` è impossibile visto che j viene eseguita prima di k in O.

continua

151

Soluzione esercizio 13 Cap. 4

Indichiamo con C'_i i tempi in cui termina la copia del cliente i nello scheduling G.
Siano k e j due richieste adiacenti in G eseguite in ordine inverso in O.
Se in G scambiamo k con j otteniamo che la somma $\sum_{i=1}^{n} w_i C'_i$ cambia come segue:

I valori C_i' delle richieste diverse dalla k e la j non cambiano
Sia C' il momento in cui viene soddisfatta la richiesta del cliente che precede k in G
Dopo aver scambiato k con j la somma $\sum_{i=1}^{n} w_i C'_i$ è modificata di una quantità pari a

$$- w_k (C'+t_k)+ w_k (C'+t_k+t_j)- w_j (C'+t_k+t_j)+w_j (C'+t_j)= w_k t_j- w_j t_k = t_j t_k (w_k / t_k - w_j /t_j)$$

Siccome $w_j/t_j \geq w_k/t_k$ allora $w_j t_k > w_k t_j$ e di conseguenza la somma $\sum_{i=1}^{n} w_i C'_i$ risulta minore o uguale del valore che aveva prima dello scambio.
Possiamo quindi scambiare in G tutte coppie adiacenti che risultano invertite rispett ad O fino a trasformare G in O. Nel fare questi scambi il valore di $\sum_{i=1}^{n} w_i C'_i$ non aumenta per cui anche O è ottimo.

152

Clustering

- Clustering. Dato un insieme U di n oggetti $p_1, \ldots, p_n$, vogliamo classificarli in gruppi coerenti
- Esempi: foto, documenti, microorganismi.
- Funzione distanza. Associa **ad ogni coppia di oggetti** un valore numerico che indica la vicinanza dei due oggetti
  - Questa funzione dipende dai criteri in base ai quali stabiliamo che due oggetti sono simili o appartengono ad una stessa categoria.
  - Esempio: numero di anni dal momento in cui due specie hanno cominciato ad evolversi in modo diverso

Problema. Dividere i punti in cluster (gruppi) in modo che punti in cluster distinti siano distanti tra di loro.
- Classificazione di documenti per la ricerca sul Web.
- Ricerca di somiglianze nei database di immagini mediche
- Classificazione di oggetti celesti in stelle, quasar, galassie.

153

153

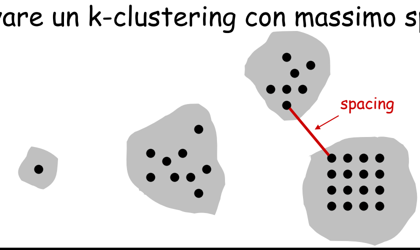
Clustering con Massimo Spacing

- k-clustering. Partizione dell'insieme U in k sottoinsiemi non vuoti (cluster).

- Funzione distanza. Soddisfa le seguenti proprietà
  - $d(p_i, p_j) = 0$ se e solo se $p_i = p_j$
  - $d(p_i, p_j) \geq 0$
  - $d(p_i, p_j) = d(p_j, p_i)$

- Spacing. Distanza più piccola tra due oggetti in cluster differenti

- Problema del clustering con massimo spacing. Dato un intero k, trovare un k-clustering con massimo spacing.



154

154

Algoritmo greedy per il clustering

- Algoritmo basato sul single-link k-clustering.
  - Costruisce un grafo sull'insieme di vertici U in modo che alla fine abbia k componenti connesse. Ogni componente connessa corrisponderà ad un cluster.
  - Inizialmente il grafo non contiene archi per cui ogni vertice u è in un cluster che contiene solo u.
  - Ad ogni passo trova i due oggetti x e y più vicini e tali che x e y sono in cluster distinti. Aggiunge un arco tra x e y.
  - Va avanti fino a che ha aggiunto n-k archi: a quel punto ci sono esattamente k cluster.

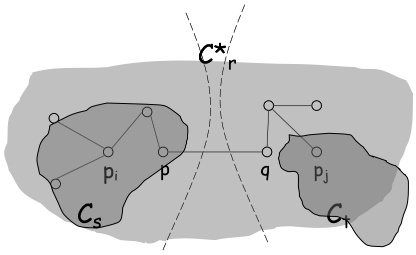
- Osservazione. Questa procedura corrisponde ad eseguire l'algoritmo di Kruskal su un grafo **completo** in cui i costi degli archi rappresentano la distanza tra due oggetti (costo dell'arco (u,v) = d(u,v)). L'unica differenza è che l'algoritmo si ferma prima di inserire i k-1 archi più costosi dello MST.
- NB: Corrisponde a cancellare i k-1 archi più costosi da un MST

155

155

Algoritmo greedy per il clustering: Analisi

- Teorema. Sia C^* il clustering $C^*_1, \ldots, C^*_k$ ottenuto cancellando i k-1 archi più costosi da un MST T del grafo completo in cui ogni arco e=(u,v) ha costo $c_e = d(u,v)$. C^* è un k-clustering con massimo spacing.
- Dim. Sia C un clustering $C_1, \ldots, C_k$ diverso da C^*
  - Sia d* lo spacing di C^*. La distanza d* corrisponde al costo del (k-1)-esimo arco più costoso dello MST T (il meno costoso tra quelli cancellati dallo MST T)
  - Facciamo vedere che lo spacing di C non è maggiore di d*
  - Siccome C e C^* sono diversi allora devono esistere due oggetti p_i e p_j che si trovano nello stesso cluster in C^* e in cluster differenti in C. Chiamiamo rispettivamente C^*_r il cluster di C^* che contiene p_i e p_j e C_s e C_t i due cluster di C contenenti p_i e p_j, rispettivamente.

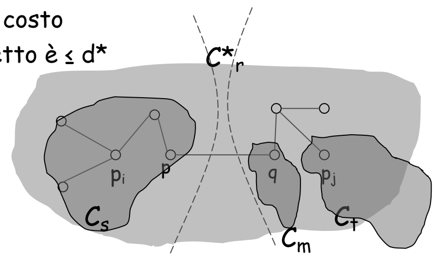


156

156

Algoritmo greedy per il clustering: Analisi

- Sia P il percorso tra p_i e p_j che passa esclusivamente per nodi di C^*_r (cioe` attraverso archi selezionati da Kruskal nei primi n-k passi) e sia q il primo vertice di P che non appartiene a C_s. Indichiamo con C_m la componente in cui si trova q.
- Sia p il predecessore di q lungo P. Il nodo p è in C_s in quanto q è il primo nodo incontrato lungo il percorso che non sta in C_s
- Tutti gli archi sul percorso P e quindi anche (p,q) hanno costo ≤ d* in quanto sono stati scelti da Kruskal nei primi n-k passi.
- Lo spacing di C è minore o uguale della distanza tra i punti piu` vicini di C_s e C_m e quindi è anche minore del costo dell'arco (p,q) che per quanto detto è ≤ d*



157

157