

Algoritmi greedy VII parte

Progettazione di Algoritmi a.a. 2019-20
Matricole congrue a 1
Docente: Annalisa De Bonis

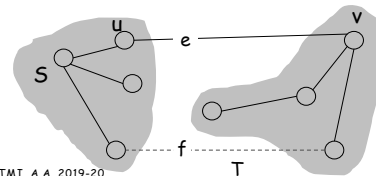
159

159

Proprietà del ciclo

- **Assumiamo che tutti i costi c_e siano distinti.**
- **Proprietà del ciclo.** Sia C un ciclo e sia $e=(u,v)$ l'arco di costo massimo tra quelli appartenenti a C . Ogni minimo albero ricoprente non contiene l'arco e .
- Dim.** (tecnica dello scambio)
 - Sia T un albero ricoprente che contiene l'arco e . Dimostriamo che T non può essere un MST.
 - Se rimuoviamo l'arco e da T disconnettiamo T in due alberi uno contenente u e l'altro contenente v . Chiamamo 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 $c_f < c_e$ allora $c(T') < c(T)$. Ne consegue che T non è uno MST.



PROGETTAZIONE DI ALGORITMI A.A. 2019-20
A. DE BONIS

160

160

Correttezza dell' algoritmo Inverti-Cancella

L' algoritmo Inverti-Cancella produce un MST.

Dim. (nel caso in cui i costi sono a due a due distinti)

Sia T il grafo prodotto da Inverti-Cancella.

Prima dimostriamo che gli archi che non sono in T non sono neanche nello MST.

- Sia e un qualsiasi arco che non appartiene a T .
- Se $e=(u,v)$ non appartiene a T vuol dire che quando l' arco $e=(u,v)$ è stato esaminato l' arco si trovava su un ciclo C (altrimenti la sua rimozione avrebbe disconnesso u e v).
- Dal momento che gli archi vengono esaminati in ordine decrescente di costo, l' arco $e=(u,v)$ ha costo massimo tra gli archi sul ciclo C .
- La proprietà del ciclo implica allora che $e=(u,v)$ non può far parte dello MST.

Abbiamo dimostrato che ogni arco dello MST appartiene anche a T . Ora dimostriamo che T non contiene altri archi oltre a quelli dello MST.

- Sia T^* lo MST. Ovviamente (V, T^*) è un grafo connesso.
- Supponiamo **per assurdo** che esista un arco (u,v) di T che non sta in T^* .
- Se agli archi di T^* aggiungiamo l' arco (u,v) , si viene a creare un ciclo. Poiché T contiene tutti gli archi di T^* e contiene anche (u,v) allora T contiene un ciclo C . Ciò è impossibile perché l' algoritmo rimuove l' arco di costo più alto su C e quindi elimina i cicli. Abbiamo quindi ottenuto una contraddizione.

161

161

Correttezza degli algoritmi quando i costi non sono distinti

- In questo caso la correttezza si dimostra perturbando di poco i costi c_e degli archi, cioè aumentando i costi degli archi in modo che valgano le seguenti tre condizioni
 1. i nuovi costi \hat{c}_e risultino a due a due distinti
 2. se $c_e < c_{e'}$ allora $\hat{c}_e < \hat{c}_{e'}$
 3. la somma dei valori aggiunti ai costi degli archi sia minore del minimo delle quantità $|c(T_1) - c(T_2)|$, dove il min è calcolato su tutte le coppie di alberi ricoprenti T_1 e T_2 tali che $c(T_1) \neq c(T_2)$ (Questo non è un algoritmo per cui non ci importa quanto tempo ci vuole a calcolare il minimo)

Continua nella prossima slide

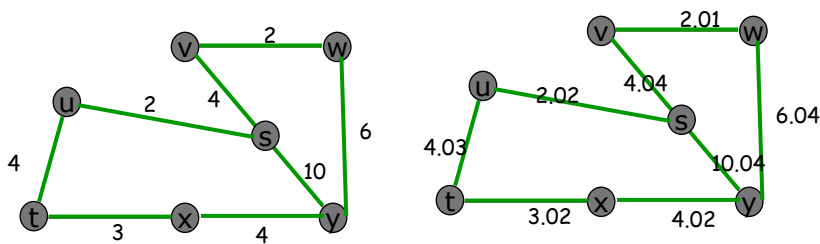
162

162

Correttezza degli algoritmi quando i costi non sono distinti

- In questo esempio i costi sono interi quindi è chiaro che i costi di due alberi ricoprenti di costo diverso differiscono almeno di 1.
- Se perturbiamo i costi come nella seconda figura, si ha che
 - I nuovi costi sono a due a due distinti
 - Se e ha costo minore di e' all'inizio allora e ha costo minore di e' anche dopo aver modificato i costi.
 - La somma dei valori aggiunti ai costi è

$$0.01+0.02+0.02+0.02+0.03+0.04+0.04 < 1$$



PROGETTAZIONE DI ALGORITMI A.A. 2019-20
A. DE BONIS

163

163

Correttezza degli algoritmi quando i costi non sono distinti

- Chiamiamo G il grafo di partenza (con i costi non perturbati) e \hat{G} quello con i costi perturbati.
- Sia T un minimo albero ricoprente del grafo \hat{G} . **Dimostriamo che T è un minimo albero ricoprente anche per G .**
- Se **per assurdo** non fosse così esisterebbe un albero T^* che in G ha costo minore di $T \rightarrow c(T) - c(T^*) > 0$, dove $c(T)$ e $c(T^*)$ sono i costi di T e T^* in G .
- Sia s la somma totale dei valori aggiunti ai costi degli archi di G
- * Per come abbiamo perturbato i costi, si ha che $c(T) - c(T^*) > s$
 - in quanto s è minore della differenza in valore assoluto tra i costi di due qualsiasi alberi ricoprenti di G .
- Se mostrassimo che il costo $\hat{c}(T^*)$ di T^* in \hat{G} è minore del costo $\hat{c}(T)$ di T in \hat{G} allora si otterrebbe una contraddizione al fatto che T è un MST per \hat{G} .

PROGETTAZIONE DI ALGORITMI A.A. 2019-20
A. DE BONIS

164

164

Correttezza degli algoritmi quando i costi non sono distinti

- Vediamo di quanto può essere cambiato il costo di T^* dopo aver perturbato gli archi. Stimiamo quindi $\hat{c}(T^*) - c(T^*)$
- Osserviamo che il costo di T^* è aumentato di un valore minore di s (perché?) $\rightarrow \hat{c}(T^*) - c(T^*) < s$

$$1. \quad \hat{c}(T^*) - c(T^*) < s \rightarrow \hat{c}(T^*) < s + c(T^*)$$

- La 1. implica

$$2. \quad \hat{c}(T) - \hat{c}(T^*) > \hat{c}(T) - c(T^*) - s > c(T) - c(T^*) - s$$

per cui la differenza tra il costo di T e quello di T^* è diminuita di un valore minore di s

Per la * si ha $c(T) - c(T^*) > s$ e quindi

$$3. \quad (c(T) - c(T^*)) - s > 0$$

La 2 e la 3 implicano $\hat{c}(T) - \hat{c}(T^*) > 0$ per cui T non può essere lo MST di \hat{G} perché T^* ha costo più piccolo di T anche in \hat{G} .

PROGETTAZIONE DI ALGORITMI A.A. 2019-20
A. DE BONIS

165

165

Correttezza degli algoritmi quando i costi non sono distinti

- **Proprietà del taglio (senza alcun vincolo sui costi degli archi)** Sia S un qualsiasi sottoinsieme di nodi e sia e un arco di costo minimo che attraversa il taglio $[S, V-S]$. Esiste un minimo albero ricoprente che contiene e .
- **Dim.**
- Siano e_1, e_2, \dots, e_p gli archi di G che attraversano il taglio ordinati in modo che $c(e_1) \leq c(e_2) \leq \dots \leq c(e_p)$ con $e_1 = e$.
- Perturbiamo i costi degli archi di G come mostrato nelle slide precedenti e facendo in modo che $\hat{c}(e_1) < \hat{c}(e_2) < \dots < \hat{c}(e_p)$. Per fare questo basta perturbare i costi c di G nel modo già descritto e stando attenti che se $c(e_i) = c(e_{i+1})$, per un certo $1 \leq i \leq p-1$, allora deve essere $\hat{c}(e_i) < \hat{c}(e_{i+1})$.
- **Sia T lo MST di \hat{G} .**
- La proprietà del taglio per grafi con costi degli archi a due a due distinti implica che lo MST di \hat{G} contiene l'arco $e \rightarrow T$ contiene e .
- Per quanto dimostrato nelle slide precedenti, T è anche un MST di G .
- **Abbiamo quindi dimostrato che esiste un MST di G che contiene e .**
- **NB: MST distinti di G potrebbero essere ottenuti permutando tra di loro archi di costo uguale nell'ordinamento $c(e_1) \leq c(e_2) \leq \dots \leq c(e_p)$**

PROGETTAZIONE DI ALGORITMI A.A. 2019-20
A. DE BONIS

166

166

Correttezza degli algoritmi quando i costi non sono distinti

- **Proprietà del ciclo (senza alcun vincolo sui costi degli archi)** Sia C un ciclo e sia e un arco di costo massimo in C . Esiste un minimo albero ricoprente che non contiene e .
- **Dim.**
- Siano e_1, e_2, \dots, e_p gli archi del ciclo C , ordinati in modo che $c(e_1) \leq c(e_2) \leq \dots \leq c(e_p)$ con $e_p = e$.
- Perturbiamo i costi degli archi di G come mostrato nelle slide precedenti e facendo in modo che $\hat{c}(e_1) < \hat{c}(e_2) < \dots < \hat{c}(e_p)$. Per fare questo basta perturbare i costi c di G nel modo già descritto e stando attenti che se $c(e_i) = c(e_{i+1})$, per un certo $1 \leq i < p-1$, allora deve essere $\hat{c}(e_i) < \hat{c}(e_{i+1})$.
- **Sia T un MST di \hat{G} .**
- La proprietà del ciclo per grafi con costi degli archi a due a due distinti implica che lo MST di \hat{G} non contiene l'arco $e \rightarrow T$ NON deve contenere e .
- Per quanto dimostrato nelle slide precedenti T è anche un MST di G .
- Abbiamo quindi dimostrato che esiste un MST di G che non contiene e .
- **NB: MST distinti di G potrebbero essere ottenuti permutando tra di loro archi di costo uguale nell'ordinamento $c(e_1) \leq c(e_2) \leq \dots \leq c(e_p)$.**

PROGETTAZIONE DI ALGORITMI A.A. 2019-20
A. DE BONIS

167

167

Correttezza degli algoritmi quando i costi non sono distinti

- Si è visto che la proprietà del taglio può essere estesa al caso in cui i costi degli archi **non** sono a due a due distinti
- Possiamo quindi dimostrare la correttezza degli algoritmi di Kruskal e di Prim nello stesso modo in cui abbiamo dimostrato la correttezza di questi algoritmi nel caso in cui gli archi hanno costi a due a due distinti.
- Si è visto che la proprietà del ciclo può essere estesa al caso in cui i costi degli archi **non** sono a due a due distinti
- Possiamo quindi dimostrare la correttezza dell'algoritmo Inverti-Cancella nello stesso modo in cui abbiamo dimostrato la correttezza dell'algoritmo nel caso in cui gli archi hanno costi a due a due distinti.

PROGETTAZIONE DI ALGORITMI A.A. 2019-20
A. DE BONIS

168

168

Clustering

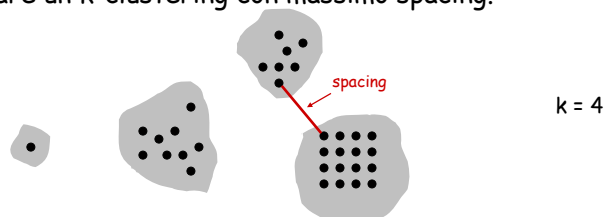
- **Clustering.** Dato un insieme U di n oggetti p_1, \dots, p_n , vogliamo classificarli in gruppi coerenti
- Esempi: foto, documenti, microorganismi.
- **Funzione distanza.** Associa ad ogni coppia di oggetti un valore numerico che indica la vicinanza dei due oggetti
 - Questa funzione dipende dai 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.

169

169

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.



170

170

Algoritmo greedy per il clustering

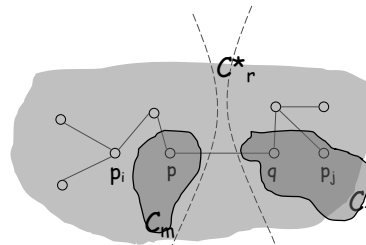
- **Algoritmo basato sul single-link k-clustering.**
 - Costruisce un grafo sull'insieme di vertici U in modo che alla fine abbia k componenti connesse. Ogni componente connessa corrisponderà ad un cluster.
 - Inizialmente il grafo non contiene archi per cui ogni vertice u è in un cluster che contiene 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

171

171

Algoritmo greedy per il clustering: Analisi

- **Teorema.** Sia C^* il clustering C^*_1, \dots, 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, \dots, 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 da T)
 - Facciamo vedere che lo spacing tra due cluster 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.

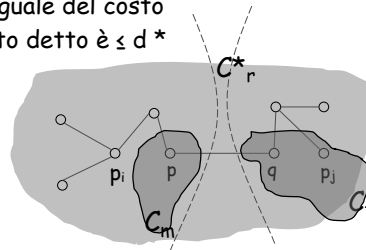


172

172

Algoritmo greedy per il clustering: Analisi

- Sia P il percorso tra p_i e p_j che passa esclusivamente per nodi di C^*_r e sia q il primo vertice di P che non appartiene a C_s (NB: deve essercene per forza uno; se non ce ne fosse uno che precede p_j allora sarebbe $q = p_j$)
- Sia p il predecessore di q lungo P . Il nodo p è in una componente C_m di C diversa da C_t , in quanto q è il primo nodo incontrato lungo il percorso che sta in C_t
- Tutti gli archi sul percorso P e quindi anche (p,q) hanno costo $\leq d^*$ in quanto sono stati scelti da Kruskal nei primi $n-k$ passi.
- Lo spacing di C è minore o uguale del costo dell'arco (p,q) che per quanto detto è $\leq d^*$



173