Domande di teoria

Problema dell'albero ricoprente di peso minimo, Kruscal e Prim-Dijkstra

1. **Illustrare il problema di determinare un albero ricoprente di costo minimo.**
2. **Dimostrare delle proprietà delle soluzioni ottime che consentono la realizzazione di algoritmi per risolvere tale problema.**
3. **Descrivere gli algoritmi noti per risolvere il problema e discuterne la complessità computazionale.**

Soluzione

1. Un albero ricoprente di un grafo connesso è un albero che collega tutti i nodi del grafo; si vuole individuare un albero ricoprente tale che la somma dei pesi degli archi appartenenti all'albero sia minimizzata.

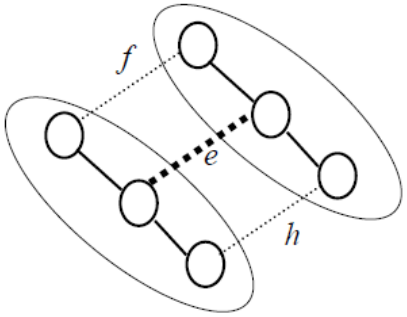
2. *Taglio fondamentale:* dato un grafo connesso ed un suo albero ricoprente si può notare che la rimozione di un qualunque arco dall'albero ricoprente generi due sottoalberi. Un albero contiene n-1 archi e quindi ci saranno n-1 tagli fondamentali. Aggiungendo un arco appartenente al taglio si ottiene nuovamente un albero ricoprente.

   Condizioni di ottimalità sui tagli: sia T* un albero ricoprente di costo minimo, un arco e apparterrà all'albero ricoprente di costo minimo se e solo se esiste un taglio fondamentale in G tale che l'arco e minimizzi il costo degli archi tagliati.
   e: archi appartenenti all'albero (ce = costo arco).
   f: archi non appartenenti all'albero (cf = costo arco).

Dimostrazione:

$cf > ce$

$ch > ce$

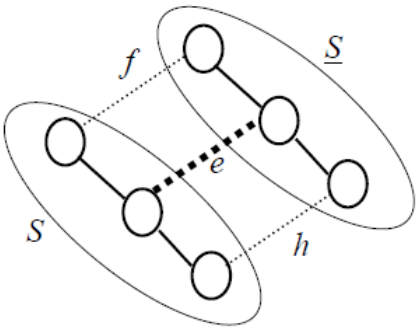


Supponiamo per assurdo che esista un arco e che minimizza un taglio, ma che tale arco non appartenga all'albero ricoprente di costo minimo. Aggiungendo all'albero ricoprente l'arco e si otterrà un ciclo fondamentale, e ci sarà almeno un altro arco che appartiene al taglio, sia questo arco f. Varrà la condizione cf > ce, quindi rimuovendo f dall'albero di costo minimo si avrebbe un albero a costo inferiore segue l'assurdo.

Dato T* dobbiamo mostrare come sia possibile costruire un taglio tale che l'arco e sia l'arco di costo minimo tra gli archi appartenenti al taglio. A tal fine consideriamo nell'insieme S una delle due componenti connesse che si verrebbe a generare a seguito della rimozione dell'arco e. L'arco e quindi appartiene ad un taglio e necessariamente sarà il minimo in quel taglio altrimenti l'albero T* non sarebbe una soluzione ottima.

$cf > ce\,?$

$ch > ce\,?$

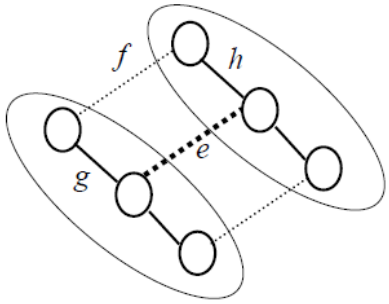


Condizioni di ottimalità sui cammini: un albero ricoprente T* è minimo se e solo se soddisfa ce <= cf per ogni arco *f* di G e per ogni arco *e* contenuto nel cammino che connette i due nodi terminali di f.

$ce <= cf$

$cg <= cf$

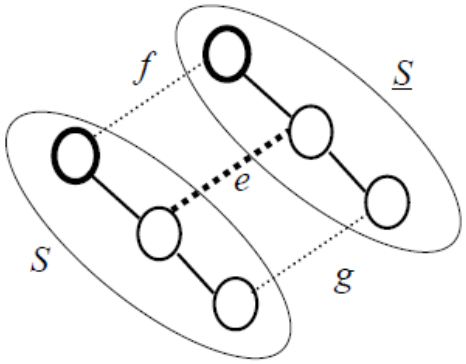
$ch <= cf$



Supponiamo per assurdo che T* sia ottimo e che esista un arco e nel cammino tra i nodi terminali dell'arco f che non soddisfi la condizione. Se vale ce >= cf allora introdurre l'arco f nell'albero ricoprente al posto dell'arco e produrrebbe un albero a costo inferiore di T*, contraddicendo l'ipotesi iniziale.

$ce <= cf$

$ce <= cg$



Siano S e $\underline{S}$ generati dal taglio legato alla rimozione di e, sia l'arco f nel taglio [S, V – S]. Visto che T* contiene un unico cammino che unisce le estremità di f, tale cammino dovrà passare per l'arco e. L'ipotesi implica ce <= cf. Considerando che questa condizione deve essere valida per ogni arco *f* nel taglio [S, S] , allora T* soddisfa le condizioni di ottimalità dei tagli da cui T* deve essere un albero a costo minimo.

3. *Algoritmo di Kruscal*
    a. Albero ricoprente vuoto.
    b. Si ordinano gli archi in ordine crescente di lista.
    c. Si scorre la lista prendendo ad ogni passo l'arco con peso minore:
        i. Se l'arco aggiunto genere un ciclo, non si aggiunge un albero ricoprente.
        ii. Se l'arco congiunge due componenti a costo minimo, apparterrà all'albero.
    d. Si ottiene un albero ricoprente di costo minimo.

La complessità dell'algoritmo è di $O(n \log(n) + n^2)$.

4. *Algoritmo di Prim-Dijkstra*
    a. Albero ricoprente vuoto.
    b. Si considera un taglio fondamentale differente.
    c. Si cerca l'arco che minimizza il peso tra gli archi appartenenti al taglio corrente.
    d. Si aggiorna il taglio corrente con l'arco selezionato nell'albero ricoprente.
    e. L'iterazione verrà ripetuta finché non si è costruito l'intero albero.

La complessità dell'algoritmo è di $O(n^3)$.

Problema di cammino minimo e Floyd-Warshall

1. Definire il problema di cammino minimo.
2. Illustrare un algoritmo noto per risolverlo nel caso in cui siano presenti archi di peso negativo.
3. Dimostrare il teorema di Floyd-Warshall.

Soluzione

1. Il problema di cammino minimo consiste nel trovare una sequenza senza ripetizioni di vertici ed archi in un grafo orientato e pesato tale che sia minimizzato il costo della sequenza.

2. Un algoritmo per risolvere il caso in cui siano presenti archi di peso negativo (ammesso sempre che non esistono cicli a peso negativo) è l'algoritmo di Floyd-Warshall.
   Enunciato: il cammino minimo da un nodo i ad un nodo j è composto dal cammino ottimo da i a k e dal cammino ottimo da k a j.
   L'algoritmo è corretto perché ad ogni iterazione aggiunge un nodo al sotto-grafo indotto considerato e valuta l'operazione triangolare a partire da quel nodo verso tutte le altre coppie di nodi già considerate; al crescere del sotto-grafo, aggiorno i cammini). Al termine avremo un ciclo a peso negativo o la matrice corretta dei costi dei cammini minimi.

3. *Dimostrazione:*
   a. L'operazione triangolare: per ogni coppia di nodi i,j si vede se per passare da i a j, conviene passare per k.
   b. Si procede per induzione:
      i. Passo base: per k=0, c[i,j] è il costo effettivo del cammino da i a j nel sotto-grafo indotto da {i,j}.

ii. Passo induttivo: supponiamo che la proprietà sia verificata all'iterazione k -1 e si considera un cammino minimo P_{ij} da i a j nel sottografo indotto da {1, …, k} U {i, j} e sia $c(P_{ij})$ il suo costo. A questo punto si possono verificare due casi:

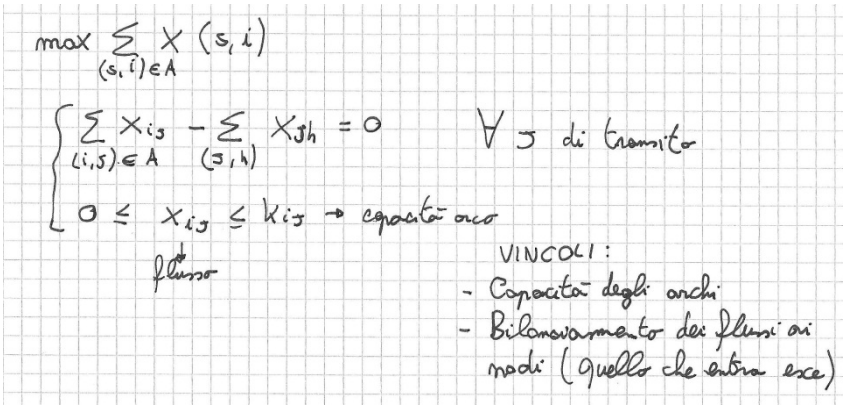
1. P_{ij} non passa dal vertice k, allora $c(P_{ij}) = c[i, j]$ per l'ipotesi induttiva;
2. P_{ij} passa dal vertice k, allora $Pij = P_{ik} U P_{kj}$.
   $c(P_{ij}) = C[i,k] + c[k,j]$ ->
   -> $c(P_{ij}) = \min \{c[i,j], c[i,k] + c[k,j]\}$
   e dopo l'operazione triangolare su k si ha che:
   $c[i,j] = c(P_{ij})$.

Problema di massimo flusso e Ford-Fulkerson

1. Definire il problema di massimo flusso.
2. Illustrare un algoritmo noto per risolverlo.
3. Dimostrare il teorema di Ford-Fulkerson.

Soluzione

1. Il problema di massimo flusso consiste nel trovare il modo migliore per trasferire entità da un punto ad un altro in una rete; ogni arco è caratterizzato da una capacità massima, ovvero da un limite massimo di entità che possono attraversarlo. Nei problemi di flusso il grafo è capacitato, ovvero gli archi hanno una portata massima che non può essere superata, mentre nei problemi di cammino minimo gli archi hanno un loro peso.



2. *Cammino aumentante:* è un cammino da S a P in cui gli archi diretti non sono saturi e gli archi inversi non sono scarichi.

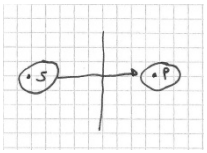
   Flusso massimo corrisponde capacità minima di taglio.

Rete residua: è un grafo con n nodi, e per ogni arco (i,J) del grafo originale definisco:

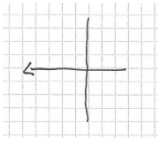
- Arco diretto: (i,j) con capacità residua $K_{ij} - X_{ij}$.
- Arco inverso: (i,j) con capacità residua X_{ij}.

Se nella rete residua c'è un cammino aumentante è possibile aumentare il flusso di una quantità pari alla minima capacità residua.

Se non esiste un cammino aumentante, esiste un taglio che separa la sorgente dal pozzo.



Tutti gli archi del taglio diretto sono saturi.



Tutti gli archi del taglio inverso sono scarichi.

3. *Dimostrazione:* per assurdo, dico che:
   il taglio non esiste -> posso raggiungere un nuovo nodo -> …
   Così via fino a raggiungere il pozzo; quanto l'ho raggiunto ho trovato un cammino aumentante.
   Ma il cammino aumentante non può esistere dal momento che ho un taglio.

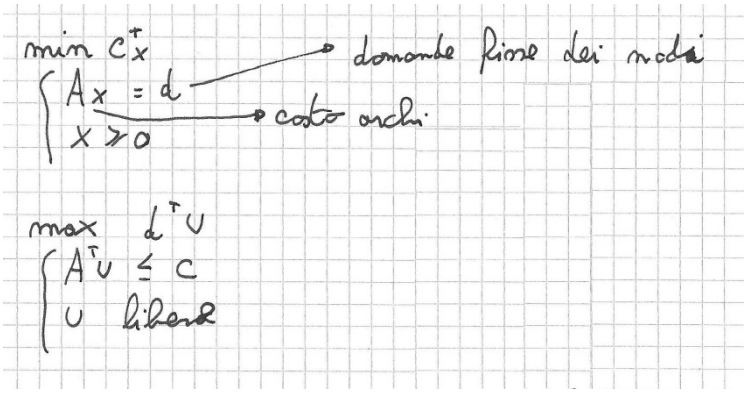
Problema di flusso di costo minimo

1. Definire il problema di flusso di costo minimo.
2. Illustrare un algoritmo noto per risolverlo.
3. Dimostrare che una base della matrice dei coefficienti del problema in forma standard corrisponde a un albero ricoprente della rete di flusso. Dimostrare la struttura grafica di una base della matrice dei coefficienti del problema in forma standard.

Soluzione

1. Il problema di flusso di costo minimo consiste nel far giungere il "prodotto" realizzato nei nodi sorgente ai nodi destinazione facendolo viaggiare attraverso la rete e cercando di spendere il meno possibile per il trasporto. Ciò che viene prodotto nei nodi sorgente è esattamente pari a ciò che viene consumato nei nodi destinazione.

2. Un algoritmo per risolverlo è il simplesso su reti.

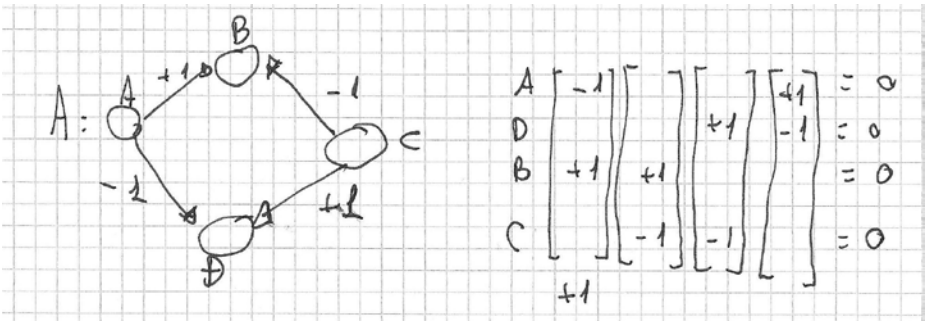


HP: il grafo è connesso con n nodi (ne potrei eliminare 1).
Base di A <-> Albero ricoprente del grado.

Imposto il duale:

a. Trovo $\underline{u}$ risolvendo: $u_j - u_i = C_{ij}$, per ogni (i,j) in base e fissando $u_1 = 0$.
b. Verifico l'ammissibilità duale di $\underline{u}$: $u_j - u_i <= C_{ij}$ per ogni (i,j) fuori base.
   Se $\underline{u}_j - \underline{u}_i > C_{ij}$ allora x_{ij} entra in base, esce l'arco discorde con il minimo flusso presente nel ciclo formato da x_{ij} entrante.

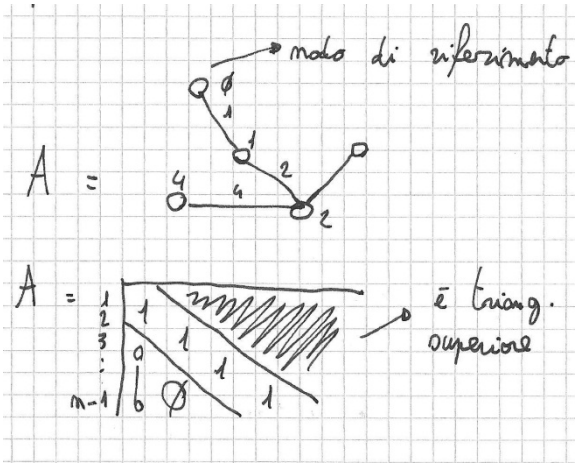
3. *Teorema:* se un grafo G è connesso, la base di A è un albero ricoprente di G.
   Dimostrazione: una base di A contiene necessariamente (n-1) colonne di A; (n-1) colonne di A che non formino una base dell'albero ricoprente devono necessariamente contenere un ciclo. Le colonne di A sono linearmente dipendenti, ovvero che esiste una combinazione lineare delle colonne con coefficienti non tutti nulli, a somma 0. Dato un ciclo, i coefficienti della combinazione lineare possono ottenersi semplicemente percorrendo il ciclo in un verso qualsiasi e fissando il coefficiente dell'arco (i,j) del ciclo:
   - 1 se l'arco è concorde al verso di percorrenza del ciclo;
   - -1 se l'arco è discorde al verso di percorrenza.



Sommando le colonne si ottiene una colonna nulla, il che dimostra che queste tre colonne sono linearmente dipendenti e quindi non possono far parte contemporaneamente di una base di A.

4. *Teorema:* un grafo G è connesso se e solo se il rango di A è uguale a n-1.

   Dimostrazione: albero ricoprente => base di A.



Sempre per le motivazioni di prima, essendo le colonne linearmente dipendenti, il rango non può essere massimo, ma solo n-1, ma dal fatto che la disponibilità totale di tutti i nodi sorgente è uguale alla domanda totale di tutti i nodi pozzo, il problema ammetterà sempre soluzione e una delle equazioni può essere cancellata.