**4.1**

Il preflusso è un vettore (flusso) x tale che i vincoli di capacità degli archi sono rispettati, mentre quelli di bilanciamento ai nodi possono non esserlo. Viene utilizzato negli algoritmi di MF per migliorarne la complessità.

**4.2**

Dato $P \subseteq R^n$ è un poliedro se e solo se esistono X, V finiti tali che
P = conv(X) + cono(V)

È importante nella programmazione lineare perché permette di definire un poliedro generato dai punti in X e dalle direzioni in V

**4.3**

Dati un insieme N={1..n} di oggetti e un insieme V={1…m} di luoghi, definiamo vincolo di semi assegnamento un unico vincolo che associa ad ogni luogo un oggetto [ somma(j=1 to m) xij=1 forAll i]

Definiamo vincoli di assegnamento invece, dei vincoli in cui ogni oggetto è assegnato ad un luogo e viceversa [somma(j=1 to m) xij=1 forAll i] and [somma(i=1 to n) xij=1 forAll j]

Nei vincoli di assegnamento |N|=|V|, mentre nei vincoli di semi assegnamento |N|<=|V|

**4.4**

Cammini minimi successivi: si basa sul concetto dello pseudoflusso, cioè un flusso che rispetta le capacità degli archi, ma che può comportare sbilanciamento positivo o negativo ad ogni nodo. Rispetta sempre 2 invarianti, ad ogni iterata x é minimale, se termina x è minimale con sbilanciamento complessivo=0. Complessità pseudopolinomiale.

Cancellazione cicli negativi: inizialmente si risolve il problema MF sul grafo e poi si cercano i cicli di costo negativo per cancellarli, poiché x è ottimale se e solo se non esistono cicli di costo negativo. La complessità è pseudopolinomiale.

**4.5**

Non abbiamo fatto Farkas.

**4.6**

Il duale si basa su una involuzione, cioè una funzione inversa di sé stessa, che mappa ogni problema di PL nel suo duale. La dualità è un concetto importante perché ci permette di giungere ad una soluzione ottima, risolvendo il problema duale.

**4.7**

Il problema di MCF è più generale perché MF è un'istanza di MCF dove i costi sono nulli, le capacità inferiori sono paria zero, il grafo ha sbilanciamenti nulli su ogni nodo, si aggiunge un arco fittizio da t ad s con capacità infinita e costo -1.

**4.8**

Teorema di decomposizione:

sia P = {x | Ax<=b} e siano X,V tali che P = conv(X)+cono(V).
Allora, attraverso il teorema di decomposizione, si può affermare che il problema max{cx|Ax<=b} ha ottimo finito se e solo se $cv_j$<=0 per ogni $v_j$€V, in tal caso esiste un k tale che $x_k$€X è soluzione ottima.

L'algoritmo del simplessi sfrutta le basi ammissibili che corrispondono ai vertici del poliedro, che sono contenuti nell'insieme X, e cerca tra le direzioni v€V quelle di crescita per massimizzare la funzione obiettivo.

**4.9**

Vedere 4.4

**4.10**

Il valore massimo del flusso coincide con la minima capacità dei tagli.

Supponiamo v = flusso massimo. Se Gx non ha cammini aumentanti esiste un taglio di capacità pari a v (siano ij* il taglio). v<= uij per ogni taglio ij (flusso minore della capacità di ogni taglio). Sapendo v=uij*, possiamo concludere che uij*<=uij per ogni taglio ij. Di conseguenza uij* è il taglio di capacità minima.

**4.11**

Definendo una variabile € N, e vincolandola ad assumere valori >=0 e <=1, cioè nell'insieme {0,1}.

Le operazioni booleane vengono modellate attraverso l'uso di operazioni matematiche lineari e attraverso l'uso di vincoli lineari.

**4.12**

Cammini minimi successivi risolve il problema MCF. La sua correttezza si basa su 2 invarianti: x è uno pseudoflusso minimale; quando l'algoritmo termina, se il vettore degli sbilanciamenti è nullo, allora x è anche il flusso di costo minimo.

**4.13**

EK è un'istanza di FF e viene introdotto per diminuire la complessità, in quanto ad ogni iterata sceglie il cammino di lunghezza minima.

**4.14**

Dati x,y soluzioni ammissibili rispettivamente per il primale e per il duale, allora cx<=by

**4.15**

Perché i vertici rappresentano le soluzioni di basi ammissibili.

**4.16**

Nei problemi di accoppiamento si lavora con grafi bipartiti non orientati, e si cerca di individuale un insieme M di archi che definiscono un accoppiamento che può essere di 3 tipi: Massima cardinalità, minimo costo, massima cardinalità bottleneck.

**4.17**

Una funzione lineare a tratti è una funzione definita da più sottofunzioni definite su intervalli diversi.

Possiamo modellare le variabili per definire l'intervallo in cui ci si trova.

**4.18**

Dati {1…m} vincoli lineari che definiscono un poliedro, dato I⊆{1…m}, k il complementare di I, indichiamo con Pi il poliedro definito da {x | Aix = bi && Akx=bk}. Se Pi è non vuoto, esso è una faccia del poliedro.

**4.19**

Non fatto

**4.20**

Una direzione di crescita è una direzione ammissibile che fa crescere la funzione obiettivo. Data E direzione di crescita, definiamo x(l) = x+lE. Di conseguenza avremo cx<= cx(l) -> cx <= cx + clE .
Una direzione E è ammissibile se esiste un valore l tale che x(l) = x +lE è ammissibile nel primale, cioè spostandoci su quella direzione per una lunghezza l si resta nel poliedro.

**4.21**

=> per contrapposizione, se esiste un ciclo negativo allora x non è ammissibile, infatti possiamo pompare flusso sul ciclo per diminuire il costo, contrastando la minimalità di x.

<= per contrapposizione supponiamo x non minimale, allora esiste un y con cy<cx con lo stesso sbilanciamento (ovvero applichiamo un ciclo) e dimostriamo il teorema con la strutta degli pseudoflussi.

**4.22**

V è uguale al flusso di ogni taglio, ed è minore o uguale alla capacità di tale taglio.

**4.23**

L'algoritmo che garantisce la complessità più bassa è Goldberg Tarjan, perché si basa su una costruzione più locale del flusso massimo, mentre in altri algoritmo con FF, EK, si tiene conto di un'analisi globale.

**4.24**

Vedi 4.4

**4.25**

1) se il primale è illimitato, il duale è vuoto.

2) se x,y sono rispettivamente soluzioni ammissibili per primale e duale, con cx=yb allora x,y sono soluzioni ottime.

**4.26**

È una descrizione astratta della parte di realtà utile al processo decisionale.

**4.27**

La definizione di vincoli lineari permette di restringere lo spazio di ricerca, che si ridurrà ai soli vertici del poliedro che sono rappresentati attraverso basi ammissibili.

**4.28**

Dato un insieme V={1…t} di direzioni, definiamo cono finitamente generato l'insieme
cono(V)={v=somma(i=1 to t) vi*ui | ui€R+}

Un cono convesso è un cono in cui, dati 2 punti qualsiasi nel cono, il segmento che unisce i due punti è contenuto nel cono.

**4.29**

Nello pseudoflusso lo sbilanciamento dei nodi può essere positivo, nullo o negativo, mentre nel preflusso lo sbilanciamento può essere solo positivo o nullo.

**4.30**

Vedi 4.7

**4.31**

Dati x,y due pseudoflussi qualunque, esistono k<=n+m cammini aumentati P1,..,Pk, con al più m cicli, tali che Z1 = x, Zk+1=y, Zi+1=Zi (+) θiPi , 0<=θ<=θ(Pi,Zi)

Tutti i Pi hanno come estremi dei nodi in cui lo sbilanciamento in x è diverso da quello in y

**4.32**

-x<=0, x<=4, -y<=0, y<=4

**4.33**

Vedi 4.3

**4.34**

Dati O+D nodi, dato Aij = 1 se il nodo i€O è connesso al nodo j€D, 0 altrimenti

Possiamo definire Xij = 1 se scelgo l'arco che collega i€O e j€D, 0 altrimenti.

La funzione obiettivo sarà max{somma(i€O)somma(j€D)xij}

Rispettando i vincoli
somma(j€D)xij<=1 per ogni i

somma(i€O)xij<=1 per ogni j

**4.35**

Già discusso nelle domande precedenti. La correttezza deriva dal fatto che un flusso è ottimo se e solo se non esistono cicli di costo negativo, e se l'algoritmo termina, non avremo cicli di costo negativo e i nodi avranno sbilanciamento nullo.

**4.36**

1) problema vuoto, l'insieme delle soluzioni ammissibili e l'insieme vuoto

2) problema illimitato, nel caso di un problema di massimo, per ogni x, esiste un g tale che cp(g)>=x, di conseguenza, la soluzione ottima è +infinito. Nei problemi di min, la soluzione ottima diventa -infinito.

3) valore ottimo finito ma non la soluzione ottima, non esiste nessuna soluzione ottima tale che il valore della funzione obiettivo in quella soluzione, sia uguale al valore ottimo

4) valore ottimo finito e soluzione ottima finita

**4.37**

L'algoritmo usato per costruire un flusso ammissibile, si basa su uno dei qualsiasi algoritmi per il problema di MF (ad es FF,EK,GT). Quindi vengono tolti i costi, le capacità inferiori, vengono aggiunti 2 nodi fittizi che fungono da unica sorgente e unico pozzo.

**4.38**

Già descritto, dimostrazione omessa palesemente non mi va di scrivere

**4.39**

Vedi 4.37

**4.40**

V = sommatoria( s-j € A) Xsj – sommatoria( i-s € A) Xis = sommatoria (k€N) [ sommatoria(k-j€A) Xkj – sommatoria(j-k€A) Xjk] = sommatoria (i-j € A+(ns-nt)) Xij – sommatoria(i-j€A-(ns-nt))Xji

V è minore o uguale poiché per ogni ij, Xij<=Uij

**4.41**

In Gx ci sono tutti i nodi. E per ogni arco ij, abbiamo un arco concorde se xij<uij con costo cij, ed un arco discorde se xij>0 con costo pari a -cij

**4.42**

Il problema MCF consiste nel trovare un flusso di costo minimo che sia ammissibili su un grafo. Se esiste, il flusso rispetta i vincoli di capacità su ogni arco, e rispetta il bilanciamento dei nodi.

**4.43**

Vedi 4.31 per definizione
È importante perché tramite esso è possibile dimostrare che un flusso è ottimo se e solo se non esistono cicli di costo negativo.

**4.44**

L'algoritmo resterebbe corretto poiché EK è una generalizzazione di FF, e quest'ultimo potrebbe prevedere ad ogni iterata cammini di lunghezza qualsiasi. Per quanto riguarda la complessità, resta invariata in quanto un cammino di lunghezza massima, potrà al più percorrere tutti gli archi, quindi O(|A|), ed ogni iterata cancella un arco critico che può tornare non critico al max N volte. Complessità O(|NA^2|).

**4.45**

1) Nozione di grafo residuo per i problemi di MF, dove per ogni arco ij, c'è un arco concorde se xij<uij, e uno discorde se xij>0

2) Grafo residuo per i pseudoflussi, dove per ogni arco ij, c'è un arco concorde con costo cij se xij<uij, e uno discorde di costo -cij se xij>0

**4.46**

E è direzione ammissibile per x se Ai(x)E<=0

Dimostrazione:

AiX(l) = AiX + lAiE<=bi

Se i€I => Aix=bi ----> lAiE<=0 ----> AiE<=0

Se i€I => basta prendere un l abbastanza piccolo

**4.47**

La complessità del simplesso è coefficiente binomiale (m n), quindi è esponenziale in m. nel caso medio, la complessità è polinomiale

**4.48**

Valore flusso ammissibile = flusso di ogni taglio <= capacità di ogni taglio

**4.49**

Algoritmo esatto -> prende in input un’istanza del problema P, e restituisce in output se esiste una soluzione ottima

Algoritmo euristico -> Calcola un’approssimazione superiore (per problemi di minimo) o inferiore (per massimo). Potrebbe concludere che non esistono soluzioni ammissibili.

**4.50**

Vedi 4.29

Preflusso in GT, pseudoflusso in cammini minimi successivi

**4.51**

Sia N={1…n} insieme di oggetti, e siano F={F1,…,Fm} insieme di sottoinsiemi di {1….n}. si vuole determinare D⊆F di costo minimo tra tutti i sottoinsiemi di F che rispettino alcuni vincoli. Rappresentiamo con aij=1 se i è nel sottoinsieme j, 0 altrimenti. Definiamo xj= 1 se j appartiene a D, 0 altrimenti.

Fo = sommatoria(j=1 to m) cj*xj

**4.52**

Con base ammissibile si intende una base di indici che rappresentano dei vincoli, tali per cui Ab è quadrata e invertibile e x=Ab^-1 * Bb (detta soluzione di base) appartiene al poliedro.

**4.53**

Un poliedro è un’intersezione di m finiti semispazi. Devono esistere A€R^m*n, X€R^n, b€R^m, tali che il poliedro P = {X | AX<=B} . Può essere caratterizzato attraverso il teorema di Motzkin e quindi rappresentato attraverso l’inviluppo convesso di un insieme X + il cono convesso generato dalle direzioni in V

**4.54**

Vedi 4.34

**4.55**

Valore ottimo è il valore ottimo della funzione obiettivo (Zp). La soluzione ottima è una soluzione g* € Valori ammissibili tale che cp(g*)=Zp

**4.56**

Il vertice è una faccia di dimensione 0.

**4.57**

Il problema duale è vuoto.

**4.58**

Massima cardinalità -> si vuole massimizzare il numero di archi facenti parte dell’accoppiamento

Minimo costo -> tra gli accoppiamenti perfetti si cerca quello di costo minimo

Massima cardinalità bottleneck -> tra tutti gli accoppiamenti di max cardinalità si cerca di minimizzare il bottleneck (definito come il massimo costo di un arco facente parte dell’accoppiamento)

**4.59**

Definizione svolta a 4.53

È cruciale perché ci permette di restringere il campo di ricerca nei problemi di minimizzazione/massimizzazione ai soli vertici del poliedro.

**4.60**

Supponiamo v={v1,…,vn} insieme di valori discreti. Creiamo n variabili logiche y1,…,yn con yi = 1 se scelgo il valore vi, 0 altrimenti.

Sommatoria(i=1 to n) yi = 1

X = sommatoria(i=1 to n) yi*vi

**4.61**

V = x(ns-nt) <= u(ns-nt)

**4.62**

Poliedro come intersezione di semispazi

Poliedro come somma di un inviluppo convesso e un cono finitamente generato

**4.63**

Vedi 4.51

**4.64**

Vedi 4-33