Appunti Ricerca Operativa

Luca Seggiani
2024

1 Lezione del 23-09-24

1.1 Introduzione

1.1.1 Programma del corso

Il corso di ricerca operativa si divide in 4 parti:

1. Modello di Programmazione Lineare;
2. Programmazione Lineare su reti, ergo programmazione lineare su grafi;
3. Programmazione Lineare intera, ergo programmazione lineare col vincolo $x \in \mathbb{Z}^n$;
4. Programmazione Non Lineare.

Le prime 3 parti hanno come prerequisiti l'algebra lineare: in particolare operazioni matriciali, prodotti scalari, sistemi lineari, teorema di Rouché-Capelli. La quarta parte richiede invece conoscenze di Analisi II.

1.1.2 Un problema di programmazione lineare

La ricerca operativa si occupa di risolvere problemi di ottimizzazione con variabili decisionali e risorse limitate. Poniamo un problema di esempio:

Problema 1.1: Produzione

Una ditta produce due prodotti: **laminato A** e **laminato B**. Ogni prodotto deve passare attraverso diversi reparti: il reparto **materie prime**, il reparto **taglio**, il reparto **finiture A** e il reparto **finiture B**. Il guadagno è rispettivamente di 8.4 e 11.2 (unità di misura irrilevante) per ogni tipo di laminato.
Ora, nel reparto materie prime, il laminato A occupa 30, ore, e lo B 20 ore. Nel reparto taglio il laminato A occupa 10 ore e lo B 20 ore. Il laminato A occupa poi 20 ore nel reparto finiture A, mentre il laminato B occupa 30 ore nel reparto finiture B. I reparti hanno a disposizione, rispettivamente, 120, 80, 62 e 105 ore. Possiamo porre queste informazioni in forma tabulare:

Reparto	**Capienza**	**Laminato A**	**Laminato B**
Materie prime	120	30	20
Taglio	80	10	20
Finiture A	62	20	/
Finiture B	105	/	30
Guadagno		8.4	11.2

Quello che ci interessa è chiaramente massimizzare il guadagno.

Decidiamo di modellizzare questa situazione con un modello matematico.

Il guadagno che abbiamo dai laminati rappresenta una **funzione obiettivo**, ovvero la funzione che vogliamo ottimizzare. Ottimizzare significa trovare il modo migliore di massimizzare o minimizzare i valori della funzione agendo sulle variabili decisionali.

La funzione obiettivo va ottimizzata rispettando determinati **vincoli**, che modellizzano il fatto che le risorse sono limitate. Una **soluzione ammissibile** è una qualsiasi soluzione che rispetta i vincoli del problema. Chiamiamo quindi **regione ammissibile** l'insieme di tutte le soluzioni ammissibili. All'interno della regione ammissibile c'è la soluzione che cerchiamo, ovvero la **soluzione ottima**.

Decidiamo quindi le **variabili decisionali**, ed esplicitiamo la funzione obiettivo e i vincoli.

In questo caso le variabili decisionali saranno le quantità di laminato A e B da produrre, che individuano un punto in $\mathbb{R}^2$ denominato (x_A, x_B). Decidere di usare la soluzione $(1, 1)$ significa decidere di produrre 1 unità di laminato A e 1 unità di laminato B, per un guadagno complessivo di $8.4 + 11.2 = 19.6$.

La funzione obiettivo sarà quindi:

$$f(x_A, x_B) = 8.4x_A + 11.2x_B, \quad f : \mathbb{R}^2 \to \mathbb{R}$$

lineare, e noi saremo interessati a:

$$\max(f(x_A, x_B))$$

rispettando i vincoli, ergo nella regione ammissibile. Per esprimere questi vincoli, cioè il tempo limitato all'interno di ogni reparto, introduciamo il sistema di disequazioni:

$$\begin{cases} 30x_A + 20x_B \leq 120 \\ 10x_A + 20x_B \leq 80 \\ 20x_A + 0x_B \leq 62 \\ 0x_A + 30x_B \leq 105 \\ -x_A \leq 0 \\ -x_B \leq 0 \end{cases}$$

dove notiamo le ultime due disequazioni indicano la positività di x_A e x_B, in forma $f(x_A, x_B) \leq b$. Questo sistema non indica altro che la regione ammissibile.

Possiamo riscrivere questo modello usando la notazione dell'algebra lineare. La funzione obiettiva e i vincoli diventano semplicemente:

$$\begin{cases} \max(c^\intercal \cdot x) \\ A \cdot x \leq b \end{cases}$$

dove c rappresenta il vettore dei costi, A rappresenta la matrice dei costi a b il vettore dei vincoli. c è trasposto per indicare prodotto fra vettori.

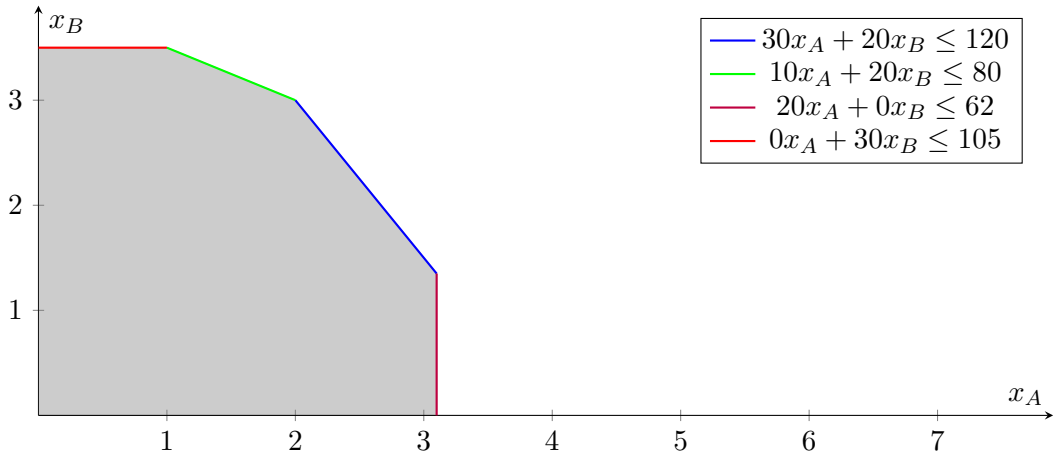
Possiamo scrivere A, b e c per esteso:

$$A : \begin{pmatrix} 30 & 20 \\ 10 & 20 \\ 20 & 0 \\ 0 & 30 \\ -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad b : \begin{pmatrix} 120 \\ 80 \\ 62 \\ 105 \\ 0 \\ 0 \end{pmatrix}, \quad c : \begin{pmatrix} 8.4 \\ 11.2 \end{pmatrix}$$

Notiamo come A e b hanno dimensione verticale $4 + 2 = 6$, dai 4 vincoli superiori e i 2 vincoli inferiori.

A questo punto, possiamo disegnare la regione ammissibile come l'intersezione dei semipiani individuati da ogni singola disuguaglianza.

Si riporta un grafico:



In diversi colori sono riportati i margini delle disequazioni, mentre in grigio è evidenziata la regione ammissibile. Qualsiasi punto all'interno della regione ammissibile vale come soluzione, e almeno uno di essi è soluzione ottimale.

Il modello finora descritto prende il nome di modello di programmazione lineare, e permette di formulare problemi di programmazione lineare (LP).

Definizione 1.1: Problema di programmazione lineare (1)

Un problema di programmazione lineare (LP) riguarda l'ottimizzazione di una funzione lineare in più variabili soggetta a vincoli di tipo $=$, $\leq$ e $\geq$, ovvero in forma:

$$\begin{cases} \min / \max(c^\intercal \cdot x) \\ A_i x \leq b \\ B_j x \geq d \\ C_k x = e \end{cases}$$

"Programmazione" qui non ha alcun legame col concetto di programmazione informatica, ma si riferisce al fatto che il modello è effettivamente *programmabile*.

"Lineare" si riferisce alla linearità dei problemi che ci permette di risolvere (e quindi del modello).

2 Lezione del 24-09-24

2.1 Forma primale standard

Ciò che abbiamo formulato finora è un problema di programmazione lineare. Possiamo dire che la forma:

$$\begin{cases} \max(c^\intercal \cdot x) \\ Ax \leq b \end{cases}$$

rappresenta un problema LP in forma **primale standard**, ricordando che c è il vettore dei coefficienti della funzione obiettivo, A la matrice dei coefficienti per ogni vincolo, e b il vettore dei termini noti per ogni vincolo.

Definizione 2.1: Forma primale standard

Un problema di programmazione lineare si dice in forma primale standard quando è espresso in forma:

$$\begin{cases} \max(c^\intercal \cdot x) \\ Ax \leq b \end{cases}$$

Si adotta una forma primale standard in quanto si può trasformare ogni problema LP in una forma di questo tipo.

2.1.1 Normalizzazione di un problema LP

Un modo per portare un problema LP qualsiasi in forma primale standard è:

1. Si trasformano le disuguaglianze: $\geq \leftrightarrow \leq$
2. Si riscrivono le uguaglianze come coppie di diseguaglianze:

$$f(x) = c \ \rightarrow \ \begin{cases} f(x) \leq c \\ f(x) \geq c \end{cases}$$

   da cui la (1):

$$f(x) = c \ \rightarrow \ S \begin{cases} f(x) \leq c \\ -f(x) \leq -c \end{cases}$$

3. Se il problema richiede il minimo, si nota che $\max(f) = -\min(-f)$, e sopratutto:

$$\bar{x} \in \operatorname{argmax}(f) \Leftrightarrow \bar{x} \in \operatorname{argmin}(-f)$$

   con $\operatorname{argmax}(f)$ e $\operatorname{argmin}(-f)$ rispettivamente gli insiemi dei punti di massimo e minimo. Questo significa che posso semplicemente cambiare di segno la funzione obiettivo per trovare da massimi minimi, e viceversa.

Notiamo inoltre che, nella forma primale standard, si ha:

$$x \in R^n, \quad A \in R^{n \times m}, \quad b \in R^m, \quad c \in R^n$$

2.2 Proprietà generali di un problema LP

La regione ammissibile di un problema PL si chiama **poliedro**. Si può dare agilmente una definizione algebrica di poliedro:

Definizione 2.2: Definizione algebrica di poliedro

Algebricamente, un poliedro è l'insieme delle soluzioni di un sistema di disequazioni lineari in $\mathbb{R}^n$ variabili:

$$P = \{x \in \mathbb{R}^n : Ax \leq b\}$$

Questa regione in un problema LP prende il nome di regione ammissibile.

Definizione 2.3: Definizione geometrica di poliedro

Geometricamente, un poliedro è l'intersezione di un numero finito di semispazi chiusi.

I semispazi chiaramente sono lineari, e in $\mathbb{R}^2$ rappresenterebbero semipiani. Chiusi significa che nelle disequazioni che descrivono i vincoli compargono solo $\leq$ e non $<$, ergo la regione ammissibile contiene la sua frontiera.

Possiamo dimostrare 4 proprietà dei poliedri:

1. Un'osservazione fondamentale è la seguente:

   Teorema 2.1: Soluzione ottimale di un problema LP

   La soluzione ottimale di un problema LP è contenuta nella frontiera della regione ammissibile.

   Questo si può ricavare dai teoremi di Fermat e di Weierstrass, e dalla convessità della regione ammissibile. Inanzitutto, si è stabilito che la soluzione ottimale non è altro che il massimo o minimo assoluto all'interno della regione ammissibile del problema. Il gradiente della funzione obiettiva è $\neq 0$ in ogni suo punto (funzione lineare a gradiente costante). Da Fermat, i massimi e minimi hanno sempre gradiente 0, ergo massimi o minimi locali (che esistono per Weierstrass) possono trovarsi solo sulla frontiera. A questo punto, possiamo imporre la convessità per asserire che quei punti di massimo o minimo sono anche globali.

2. Prendiamo in esempio il poliedro dato da:

   $$\begin{cases} x_A > 0 \\ x_B > 0 \end{cases}$$

   o se vogliamo, in forma primale standard, dato dalle matrici A e b:

   $$A : \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad b : \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

   questo poliedro non è limitato nella direzione positiva, ergo può arrivare a valori di x_A e x_B che tendono a $+\infty$. Da ciò si ha che può accadere che un problema LP ammetta soluzioni x tali che $x \to \pm\infty$, ovvero che il poliedro sia illimitato. In particolare, un poliedro limitato si dice **politopo**.

3. Notiamo poi che la soluzione di un problema LP può non essere unica. Questo accade ad esempio quando la soluzione sta su un segmento di frontiera: a quel punto tutti i punti del segmento sono soluzione. Da questo segue che:

   Teorema 2.2: Unicità della soluzione ottimale di un problema LP

   Se un problema LP ha almeno 2 soluzioni, allora ne ha infinite.

Ciò si può dimostrare come segue. Si riporta innanzitutto la notazione parametrica del segmento $z\bar{w}$, dati i due vettori di estremo z e w:

$$\lambda z + (1-\lambda)w, \quad \lambda \in [0,1]$$

A questo punto si pone che z e w sono entrambi soluzioni ottime, ergo:

$$\max(c^\intercal \cdot x) = c^\intercal z = c^\intercal w = v$$

da cui si può dire che:

$$c^\intercal(\lambda z + (1-\lambda)w) = \lambda c^\intercal z + (1-\lambda)c^\intercal w = \lambda v + (1-\lambda)v = v$$

Ovvero ogni punto sul segmento porta la funzione obiettiva a massimo assoluto, quindi è soluzione ottimale.

4. Infine, notiamo che il poliedro della regione ammissible di un problema LP può essere vuoto, ergo $P = \emptyset$. In questo caso, si ha che $\max(c^\intercal \cdot x) = -\infty$ e $\min(c^\intercal \cdot x) = \infty$. Un poliedro vuoto significa che i vincoli stessi vanno modificati. Questo solitamente si fa risovendo una versione semplificata del problema LP.

Si può fare un'altro esempio per sottolineare l'importanza del punto di massimo (o minimo), e non quel massimo (o minimo). Finché nella funzione obiettivo i coefficienti compargono nello stesso rapporto (ergo finché si scelgono vettori c linearmente dipendenti), il punto di massimo (o minimo) non cambia, per via della linearità (e si presume omogeneità) della funzione obiettiva stessa. Sarà solo il massimo (o minimo) a variare di un rapporto pari a quello di cui variano i coefficienti.

2.3 Gradiente e linee di isocosto

Si può dimostrare il seguente teorema:

Teorema 2.3: Gradiente della funzione obiettivo

Il gradiente di una funzione obiettivo definita come $f(x) = c^\intercal \cdot x$ sulla base di un qualche vettore c è in ogni punto il vettore c stesso.

Da questo gradiente si possono ricavare le cosiddette linee di isocosto (in dimensioni > 2 sarebbero superfici), cioè linee a valore costante della funzione obiettivo.

Definizione 2.4: Linea di isocosto

Si definisce linea di isocosto di una funzione obiettivo con vettore c una retta (o superficie):

$$f(x) = c^\intercal \cdot x = k$$

per un qualsiasi k costante.

2.4 Cono di competenza

Dovrebbe essere chiaro adesso che i punti di soluzione ottima stanno tutti su un segmento o su un punto della frontiera. Nel caso si abbia un vettore gradiente perpendicolare

ad un segmento della frontiera, quel segmento sarà soluzione ottima. In caso contrario, spostandoci a destra avremo l'estremo destro del segmento, e spostandoci a sinistra viceversa, finché non si diventerà perpendicolari a qualche altro segmento di frontiera.

Il cono (in R^2, angolo) di valori possibili del gradiente che rendono un punto ottimale prende il nome di **cono di competenza**.

Definizione 2.5: Cono di competenza

Il cono di competenza di un punto x^* è il cono, ovvero l'insieme di vettori gradiente, tale per cui il punto x^* conserva l'ottimalità sulla funzione obiettivo coi vincoli imposti.

Vedremo in seguito l'importanza di una nozione di cono per i problemi LP.

3 Lezione del 25-09-24

3.1 Assegnamento di costo minimo

Vediamo un problema:

Problema 3.1: Assegnamento

Quattro agenzie possono occuparsi di 4 progetti. Ogni agenzia presenta il costo stimato per la realizzazione di ogni progetto, in migliaia di euro. In forma tabulare, si riportano i valori:

	Agenzia 1	**Agenzia 2**	**Agenzia 3**	**Agenzia 4**
Progetto 1	20	17	16	14
Progetto 2	22	16	19	15
Progetto 3	21	17	15	23
Progetto 4	19	18	14	24

Vogliamo capire quale agenzia deve occuparsi di quale progetto per minimizzare i costi.

Con n agenzie e progetti ci sono $n!$ possibili combinazioni, ergo dobbiamo trovare un algoritmo efficiente. Applicando il modello studiato finora, abbiamo la matrice dei costi c:

$$\begin{pmatrix} 20 & 17 & 16 & 14 \\ 22 & 16 & 19 & 15 \\ 21 & 17 & 15 & 23 \\ 19 & 18 & 14 & 24 \end{pmatrix}$$

che possiamo portare a:

$$c : (-18, +18 + ... + 24)$$

come linearizzazione lessicografica della tabella sopra riportata (notare che sarebbe un vettore colonna).

Adesso dobbiamo solo trovare un metodo per esplicitare i vincoli del problema:

- Ogni agenzia può occuparsi solo di un progetto;

- Ogni progetto richiede solo un'agenzia.

Possiamo rappresentare la corrispondenza fra elementi come un grafo, e quindi riportarne una matrice d'adiacenza. Assumendo di appaiare elementi con lo stesso numero, avremo:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

La caratteristica di questa matrice, chiamiamola x, e che ogni elemento x_{ij} è:

$$x_{ij} = \begin{cases} 0 \\ 1 \end{cases}$$

Decidiamo di trattare la x come un vettore linearizzato lessicograficamente dalla matrice, proprio come avevamo fatto per il vettore costo. Per una matrice di adiacenza $n \times n$, di n elementi in ogni categoria, questo vettore ha dimensione n^2. Questo semplifica la notazione del problema, e sopratutto della matrice A, che sarebbe lasciando x matrice effettivamente un tensore.

Si dimostra quindi facilmente che i vincoli riportati prima possono quindi esprimersi come:

$$\begin{cases} x_{11} + x_{12} + x_{13} + x_{14} = 1 \\ x_{21} + x_{22} + x_{23} + x_{24} = 1 \\ x_{31} + x_{32} + x_{33} + x_{34} = 1 \\ x_{41} + x_{42} + x_{43} + x_{44} = 1 \end{cases}$$

per il primo punto, e:

$$\begin{cases} x_{11} + x_{21} + x_{31} + x_{41} = 1 \\ x_{12} + x_{22} + x_{32} + x_{42} = 1 \\ x_{13} + x_{23} + x_{33} + x_{43} = 1 \\ x_{14} + x_{24} + x_{34} + x_{44} = 1 \end{cases}$$

per il secondo. Imponendo la positività, si hanno quindi le matrici A e b:

$$A : \begin{pmatrix} 1 & 1 & 1 & 1 & 0... & & & & & ...0 \\ 0... & & ...0 & 1 & 1 & 1 & 1 & 0... & & ...0 \\ ... & & & & & & & & & \\ 0... & & & & & ...0 & 1 & 1 & 1 & 1 \\ ... & & & & & & & & & \\ -1 & 0... & & & & & & & & ...0 \\ ... & & & & & & & & & \\ 0... & & & & & & & & & -1 \end{pmatrix}, \quad b : (1, ..., 1, 0, ..., 0)$$

Si nota che il numero di vincoli necessari per n elementi è $2n + n^2$.

3.1.1 Assegnamento cooperativo e non cooperativo

A questo punto conviene fare una distinzione. Abbiamo definito finora il modello:

$$\begin{cases} \min c^\intercal \cdot x \\ x_{11} + x_{12} + x_{13} + x_{14} = 1 \\ ... \\ x_{41} + x_{42} + x_{43} + x_{44} = 1 \\ x_{11} + x_{21} + x_{31} + x_{41} = 1 \\ ... \\ x_{14} + x_{24} + x_{34} + x_{44} = 1 \end{cases}$$

che così scritto non nega la possibilità di x con componenti reali. Nell'esempio ciò significa sono ammesse soluzioni dove più agenzie danno contributi reali ai progetti, che possiamo semanticamente interpretare come condividere il carico di lavoro, pur rispettando i vincoli imposti. Decidiamo che questo è corretto se si parla di un problema di **assegnamento cooperativo**. Visto che il problema posto non era di questo tipo, ma era di **assegnamento non cooperativo**, si introduce un'ulteriore vincolo:

$$x \in \mathbb{Z}^n$$

Adesso ogni azienda darà un contributo intero al suo progetto, ergo coi vincoli imposti prima, ogni azienda sarà unica nel dirigere un solo progetto.

Più formalmente, possiamo dire che il passaggio ad assegnamento cooperativo comporta un **rilassamento** dei vincoli del problema. Ovvero, in generale, se un problema non cooperativo ha minimo ottimale nc, il suo associato cooperativo avrà minimo ottimale c con:

$$c \leq nc$$

3.1.2 Forma primale standard

Portiamo quindi questo problema in una forma primale simile a quella vista per altri problemi LP, concesso il vincolo $x \in \mathbb{Z}^n$.

Finora avevamo usato le trasformazioni equivalenti per problemi LP:

1. Trasformazione delle disuguaglianze: $\geq \leftrightarrow \leq$
2. Trasformazione delle uguaglianze:

$$f(x) = c \ \rightarrow \ \begin{cases} f(x) \leq c \\ -f(x) \leq -c \end{cases}$$

3. Trasformazione minimo / massimo:

$$\max f = -\min f \text{ e sopratutto } \bar{x} \in \operatorname{argmax}(f) \Leftrightarrow \bar{x} \in \operatorname{argmin}(-f)$$

Possiamo applicare queste trasformazioni al modello, in particolare la (2), che porta il numero di vincoli a $4n + n^2$.

3.2 Introduzione di surplus

Vediamo un ulteriore tecnica per trasformare problemi LP: si può portare una disequazione del tipo:

$$a_1x_1 + a_2x_2 + ... + a_nx_n \leq b$$

in un uguaglianza introducendo una variabile ausiliaria s:

$$a_1x_1 + a_2x_2 + ... + a_nx_n + s = b$$

s prende il nome di **slack**, in italiano scarto, o *surplus*.

4 Lezione del 26-09-24

4.1 Geometria dei poliedri

Introduciamo progressivamente i tipi di **combinazione** che ci sono utili nello studio dei problemi di programmazione lineare.

4.1.1 Combinazioni lineari

Definizione 4.1: Combinazione lineare

Dati $x_1, x_2, ..., x_k \in \mathbb{R}^n$ punti, y si dice **combinazione lineare** di $x_1, x_2, ..., x_k$ se:

$$\exists \lambda_i \quad (i = 1, ..., k) \quad \text{t.c.} \quad y = \sum_{i=1}^{k} \lambda_i x_i$$

Le combinazioni lineari sono utili per esprimere la funzione obiettiva sulla base dei vettori costo, ma non bastano a trovarne una soluzione ottimale.

4.1.2 Combinazioni convesse

Si introduce quindi il concetto di:

Definizione 4.2: Combinazione convessa

Dati $x_1, x_2, ..., x_k \in \mathbb{R}^n$ punti, y si dice **combinazione convessa** di $x_1, x_2, ..., x_k$ se:

$$\exists \lambda_i \in [0, 1] \quad (i = 1, ..., k), \quad \sum_{i=1}^{k} \lambda_i = 1 \quad \text{t.c.} \quad y = \sum_{i=1}^{k} \lambda_i x_i$$

Possiamo dare un esempio di cos'è la combinazione convessa di due punti in $\mathbb{R}^2$. Posti x_1 e x_2, si ha:

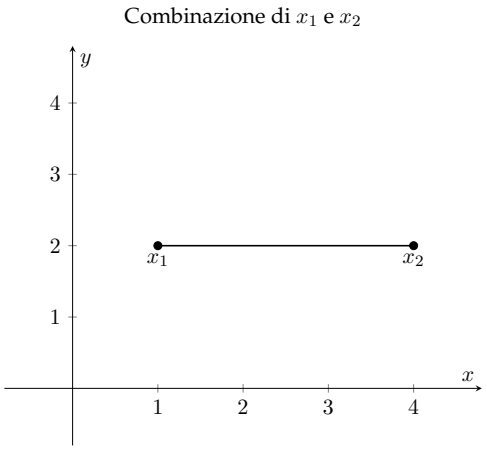
$$\lambda_1 + \lambda_2 = 1 \Rightarrow \lambda_2 = (1 - \lambda_1), \quad y = \lambda x_1 + (1 - \lambda)x_2, \quad \lambda \in [0, 1]$$

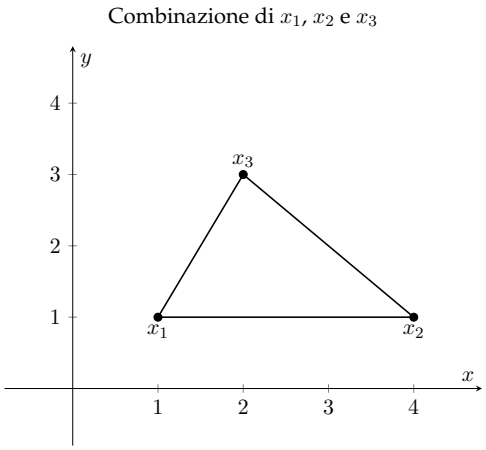
che riconosciamo essere l'equazione di un segmento $x_1\bar{x}_2$ (primo grafico).

Possiamo provare con tre punti: si avrà:

$$y = \lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3, \quad \lambda_1 + \lambda_2 + \lambda_3 = 1, \quad \lambda_i \in [0, 1]$$

che si riconduce all'equazione del triangolo di vertici x_1, x_2, x_3 (secondo grafico).





Dai grafici si nota come una combinazione convessa descrive una parte di spazio, che si può definire:

Definizione 4.3: Involucro convesso

L'involucro convesso $\text{conv}(K)$ di un'insieme di punti $K = \{x_1, x_2, ..., x_n\}, \quad x_i \in \mathbb{R}^n$ è definito come il luogo di tutte le loro combinazioni convesse.

Si nota che l'involucro convesso negli esempi precedenti è effettivamente un poliedro convesso che contiene tutti i punti che lo formano. Si può infatti dire:

Teorema 4.1: Minimalità dell'involucro convesso

L'insieme $\text{conv}(K)$ di tutte le combinazioni convesse di n punti è l'insieme convesso minimale che li contiene tutti.

In R^n, possiamo esprimere un insieme convesso lineare come un poliedro convesso, e quindi dire che l'involucro convesso di n punti è il più piccolo (in termini di inclusione) poliedro convesso che li contiene tutti. Inoltre, un'insieme è convesso se e solo se corrisponde al suo involucro convesso.

Si noti che non è detto che a n punti corrisponda un poligono di n vertici. Potrebbe infatti accadere che uno dei punti (chiamiamolo x_j) sia già parte dell'involucro convesso, ergo $x_j \in \text{conv}(K)$.

Le combinazioni convesse ci permettono di descrivere parte delle regioni ammissibili (poliedri) dei problemi di programmazione lineare, ma restano ancora in sospeso problemi che ammettono regioni illimitate. Per descrivere tali regioni, si introduce un altro tipo di combinazione.

4.1.3 Combinazioni coniche

Definizione 4.4: Combinazione conica

Dati $x_1, x_2, ..., x_k \in \mathbb{R}^n$ punti, y si dice **combinazione conica** di $x_1, x_2, ..., x_k$ se:

$$\exists \lambda_i \geq 0 \quad (i = 1, ..., k) \quad \text{t.c.} \quad y = \sum_{i=1}^{k} \lambda_i x_i$$

La combinazione conica di più punti non è più il poliedro convesso che li contiene, ma il cono con vertice nell'origine, convesso o meno, che li contiene, definito come:

Definizione 4.5: Involucro conico

L'involucro conico $\text{cono}(K)$ di un'insieme di punti $K = \{x_1, x_2, ..., x_n\} \in \mathbb{R}^n$ è definito come il luogo di tutte le loro combinazioni coniche.

Questo cono si estende fino all'infinito ($\lambda_i \geq 0$) nelle direzioni dei vettori che lo formano. Il concetto è simile a quello di spazio somma, ma con la differenza che non si va ovunque nello span dei due vettori, ma si seguono le semirette che essi conducono.

Si può dire, analogamente alle combinazioni convesse, che il cono di n punti è il pi+ piccolo (in termini di inclusione) cono convesso che li contiene. Inoltre, un'insieme è un cono convesso se e solo se corrisponde al suo involucro conico.

4.2 Poliedri

Abbiamo definito un poliedro come la regione definita da un sistema di disequazioni lineari, o geometricamente come l'intersezione di un numero finito di semipiani chiusi in $\mathbb{R}^n$. Si dimostra che un poliedro, in quanto intersezione di insiemi convessi, è lui stesso convesso: potremo applicare la definizione di convessità per dire che, presi due punti $x_1, x_2 \in C$ nell'intersezione $C = C_1 \cap C_2$, si avrà che entrambi appartengono sia a C_1 che a C_2, ergo il segmento che li congiunge completamente in ciascuno di quei due insiemi è interamente contenuto in C.

Un poliedro che è anche cono si chiama cono poliedrico. Si dimostra che:

Teorema 4.2: Cono poliedrico

Se P è un cono poliedrico allora:

$$\exists Q \quad \text{t.c.} \quad P = \{x \in \mathbb{R}^n : Qx \leq 0\}$$

con Q matrice.

Senza dimostrazioni, questo è chiaro dal fatto che le disequazioni che compongono il poliedrico sono omogenee (hanno frontiere che passano dall'origine).

Si definiscono poi i **vertici** del poliedro:

Definizione 4.6: Vertice

Un vertice di un poliedro è un punto che non si può esprimere come combinazione convessa propria di altri punti del poliedro. Si indica l'insieme dei vettori di un poliedro P come $\text{vert}(P)$.

Notiamo che i vertici di un poliedro limitato corrispondono ai punti che formano la combinazione convessa equivalente al poliedro. Per poliedri illimitati, introduciamo invece:

Definizione 4.7: Direzione di recessione

Un vettore d è la direzione di recessione di un poliedro se:

$$x + \lambda d \in P \quad \forall x \in P, \quad \forall \lambda \geq 0$$

Si indica come $\text{rec}(P)$ l'insieme delle direzioni di recessione di un poliedro.

Chiaramente, per ogni poliedro P, $0 \in \text{rec}(P)$ e per i poliedri limitati, $\text{rec}(P) = \{0\}$. Notiamo che le direzioni di recessione determinano i vettori del cono che coincide (almeno a distanze abbastanza grandi dall'origine) con i poliedri illimitati.

Inoltre, l'insieme $\text{rec}(p)$ di un poliedro è in sé un cono poliedrico. In particolare:

Teorema 4.3: Cono di recessione

Il cono di recessione di un poliedro P è dato da:

$$P = \{x \in \mathbb{R}^n : Ax \leq b\} \Rightarrow \text{rec}(P) = \{x \in \mathbb{R}^n : Ax \leq 0\}$$

Dimostrazione

Questo teorema vale perchè, per definizione, una direzione di recessione è $x + \lambda d$, che sostituita al sistema dà:

$$A(x + \lambda d) \leq b \Rightarrow Ax + \lambda Ad \leq b$$

Possiamo quindi sottrarre Ax da entrambi i lati:

$$\lambda Ad \leq b - Ax$$

Da ipotesi, $Ax \leq b$ e quindi $b - Ax \geq 0$. Chiamiamo questa quantità postiva k^+ e scriviamo:

$$\lambda Ad \leq k^+$$

Visto che $\lambda \to +\infty$, dovrà essere verò che $Ad \leq 0$, ergo con $d = x$, il teorema è dimostrato.

Un'altro teorema importante lega il cono ai suoi "raggi estremi", cioè quell'insieme finito di vettori la cui combinazione conica corrisponde all'intero cono:

Teorema 4.4: Raggi estremi

Un cono poliedrico $P = \{x \in \mathbb{R}^\times : Ax \leq 0\}$ è l'involucro conico di un'insieme limitato dei suoi punti. Questi punti prendono il nome di *raggi estremi*.

Dimostrazione

Diamo una dimostrazione più o meno intuitiva di questo risultato. Indichiamo con $A_1, ..., A_m$ le righe della matrice A. Queste formeranno dei vettori che indicano la direzione su cui "agisce" la diseguaglianza. Si può semplicemente osservare che questi vettori sono perpendicolari agli estremi del cono. Prendiamo quindi il cono dei vettori, che formerà un'altro cono detto *duale* del cono di partenza:

$$\text{cono}(A_1, ..., A_m) = \{x \in \mathbb{R}^n : Qx \leq 0\}$$

I vettori $Q_1, ..., Q_t$, presi come righe della matrice, sono perpendicolari ai vettori $A_1, ..., A_n$ di partenza, ergo paralleli ai raggi estremi del cono di partenza. Possiamo quindi dire che:

$$P = \text{cono}(Q_1, ..., Q_t)$$

4.2.1 Spazio di linealità

Definiamo infine lo spazio di linealità:

Definizione 4.8: Spazio di linealità

Lo spazio di linealità di un poliedro illimitato P è il più piccolo sottospazio contenuto interamente in P.

Ogni vettore di base di uno spazio di linealità è un vettore d tale che:

$$d \in \text{rec}(P), \quad -d \in \text{rec}(P)$$

ovvero un vettore che è contenuto sia positivo che negativo nelle direzioni di recessione del poliedro.

Questa distinzione è importante in quanto non si può dimostrare completamente il prossimo teorema su poliedri con spazio di linealità $\neq 0$.

4.2.2 Teorema di rappresentazione dei poliedri

Gli strumenti che abbiamo stabilito finora ci permettono di enunciare un'importante risultato, noto come **teorema di rappresentazione dei poliedri**, o teorema di Minkowski-Weyl.

Teorema 4.5: Rappresentazione dei poliedri

Dato un poliedro P definito come $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, si ha:

$$\exists V = \{v_1, ..., v_k\} \in \text{vert}(P), \quad \exists E = e_1, ..., e_p \in \text{rec}(P) \quad \text{t.c.} \quad P = \text{conv}(V) + \text{cono}(E)$$

Questo significa che è possibile rappresentare qualsiasi poliedro attraverso i suoi vertici, e le direzioni in cui si estende all'infinito (ergo le sue direzioni di recessione).

4.2.3 H-rappresentazioni e V-rappresentazioni

Possiamo intendere il risultato precedente come segue: per ogni poliedro di un problema di programmazione LP, abbiamo due possibili rappresentazioni:

- **H-rappresentazione:** come intersezione di semispazi, ergo come insieme delle soluzioni di un sistema di disequazioni lineari.
- **V-rappresentazione:** come un'insieme di vertici e direzioni di regressione, quindi un involucro convesso e un cono di regressione.

Non diamo una dimostrazione del teorema ma possiamo, noti i concetti di H-rappresentazione e V-rappresentazione, dire che:

- Da un **H-rappresentazione** si può ricavare una **V-rappresentazione**: abbiamo un poliedro dato dall'H-rappresentazione $Ax \leq b$. Dividiamo questo poliedro in una parte limitata e una parte illimitata: la parte limitata sarà data dai vertici, mentre la parte illimitata sarà data dal cono di regressione:

  $$P = \text{conv}(v_1, ..., v_k) + rec(P)$$

  A questo punto possiamo esprimere, come dallo scorso lemma, come combinazione conica dei suoi raggi estremi:

  $$P = \text{conv}(v_1, ..., v_k) + \text{cono}(e_1, ..., e_p)$$

- Da un **V-rappresentazione** si può ricavare una **H-rappresentazione**: abbiamo un poliedro dato dalla V-rappresentazione $P = \text{conv}(v_1, ..., v_k) + \text{cone}(e_1, ..., e_p)$. Prendiamo separatamente gli involucri convessi e conici: avremo che per l'involucro convesso possiamo immediatamente riportare in una forma ad intersezione di semipiani (disequazioni):

  $$\text{conv}(v_1, ..., v_k) = \{x \in \mathbb{R}^n : A'x \leq b'\}$$

  mentre per l'involucro conico possiamo portare in una forma ad intersezione di disequazioni omogenee:

  $$\text{cone}(e_1, ..., e_p) = \{x \in \mathbb{R}^n : A''x \leq 0\}$$

  Per trovare l'H-rappresentazione, basterà combinare queste due rappresentazioni in una forma del tipo:

  $$P = \{x \in \mathbb{R}^n : A'x \leq b', A''x \leq 0\}$$

La somma in $P = \text{conv}(V) + \text{cono}(E)$ si riferisce alla somma vettoriale fra tutti i possibili punti di $\text{conv}(V)$ e $\text{conv}(E)$, come quella studiata sui sottospazi vettoriali (anche se nessuno dei due insiemi è un sottospazio vettoriale). Per un dato insieme $\text{conv}(V)$, quindi, l'aggiunta di $\text{cono}(E)$ rappresenta la "proiezione" di tale insieme nelle direzioni di recessione indicate dal cono.

Più propriamente, posto $\text{lineal}(P) = 0$, si ha:

Teorema 4.6: Rappresentazione dei poliedri non lineali

Dato un poliedro P definito come $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, tale che $\text{lineal}(P)$, si ha:

$$P = \text{conv}(\text{vert}(P)) + \text{rec}(P)$$

la limitazione di linealità è necessaria in quanto un poliedro lineale potrebbe non essere rappresentato, nelle sue dimensioni infinite, dal semplice insieme dei suoi vettori. In verità, è possibile dimostrare che:

Teorema 4.7: Linealità di poliedri

Per ogni poliedro P non vuoto si ha:

$$\text{vert}(P) \neq \emptyset \Leftrightarrow \text{lineal}(P) = 0$$

ergo applicando lo scorso teorema potremmo provare a rappresentare un poliedro attraverso un'insieme di vettori vuoto.

Per i poliedri che otteniamo dai problemi di programmazione lineare, però, abbiamo i corollari:

- Un poliedro limitato è l'involucro convesso dei suoi vertici;
- Se il poliedro ha vincoli di positività sulle sue variabili, allora non è lineale, ergo si applica il teorema di rappresentazione. Questo è il tipo di poliedri a cui siamo abituati.

4.3 Teorema fondamentale della PL

Quanto riportare finora sulla geometria dei poliedri può essere usato per dimostrare il seguente teorema:

Teorema 4.8: Teorema fondamentale della PL

Sia dato un poliedro P rappresentato come:

$$P = \text{conv}(V) + \text{cono}(E), \quad V = \{v_1, ..., v_k\}, \quad E = \{e_1, ..., e_p\}$$

Se il problema $\mathcal{P}$ con regione ammissibile P ha valore ottimo finito, allora esiste $s \in \{1, ..., k\}$ tale che v_k è soluzione ottima di $\mathcal{P}$.

In sostanza, se un problema LP ha soluzione, essa si trova su uno dei vertici del poliedro della regione ammissibile.

Dimostrazione

Sia dato un problema LP $\mathcal{P}$ in forma primale standard, ergo posto come:

$$\begin{cases} \max c^\intercal \cdot x \\ Ax \leq b \end{cases}$$

ergo con regione ammissibile rappresentata da un poliedro P.

Dal teorema della rappresentazione, possiamo esprimere il poliedro come:

$$P = \text{conv}(V) + \text{cono}(E), \quad V = \{v_1, ..., v_k\}, \quad E = \{e_1, ..., e_p\}$$

Combiniamo le due equazioni, ergo esprimiamo prima il punto $\bar{x}$ generico del poliedro applicando le definizioni di involucro convesso e conico:

$$\bar{x} \in P : P = \text{conv}(V) + \text{cono}(E), \quad \bar{x} = \sum_{i=1}^{k} \lambda_i v_i + \sum_{j=i}^{p} \mu_j e_j$$

ed esprimiamo quindi la funzione obiettivo come il prodotto scalare fra il vettore costo e il punto $\bar{x}$ del poliedro:

$$c^\intercal \cdot \bar{x} = c^\intercal \cdot \left(\sum_{i=1}^{k} \lambda_i v_i + \sum_{j=i}^{p} \mu_j e_j \right) = \sum_{i=1}^{k} \lambda_i c^\intercal v_i + \sum_{j=i}^{p} \mu_j c^\intercal e_j$$

A questo punto conviene chiarire su cosa significa che il problema ha valore ottimo finito. Il secondo termine è la combinazione conica della sommatoria dei vettori di

recessione scalati dal vettore costo. Se almeno uno dei $c^\intercal e_j > 0$, si avrà che portando $\mu_j \to +\infty$ la funzione avrà massimo $= \infty$. Geometricamente, questo significa che esiste una direzione illimitata del poliedro dove i vettori costo permettono alla funzione di crescere all'infinito.

Dunque sarà vero che $c^\intercal e_j \leq 0 \quad \forall j \in \{1, ..., p\}$ se vogliamo che la funzione abbia valore ottimo finito.

Possiamo quindi usare questa ipotesi per dire:

$$c^\intercal \cdot \bar{x} = \sum_{i=1}^{k} \lambda_i c^\intercal v_i + \sum_{j=i}^{p} \mu_j c^\intercal e_j \leq \sum_{i=1}^{k} \lambda_i c^\intercal v_i \leq \sum_{i=1}^{k} \max_{1 \leq i \leq p} (\lambda_i c^\intercal v_i)$$

$$= \left(\max_{1 \leq i \leq p} c^\intercal v_i \right) \sum_{i=1}^{k} \lambda_i = \max_{1 \leq i \leq p} c^\intercal v_i = c^\intercal v_k$$

E quindi $\max_{x \in P} c^\intercal \cdot x \leq c^\intercal v_k$. A questo punto, visto che $\bar{x}$ è effettivamente un punto della regione ammissibile, sarà vero che:

$$c^\intercal v_k \leq \max_{x \in P} c^\intercal \cdot x$$

E dunque $\max_{x \in P} c^\intercal \cdot x = c^\intercal v_k$, come volevasi dimostrare.

5 Lezione del 30-09-24

5.1 Calcolo dei vertici

Troviamo adesso il modo di calcolare i vertici del poliedro di un problema LP in forma primale standard.

Avevamo già dato la definizione di vertice come punto non ottenibile come combinazione convessa degli altri punti del poliedro (definizione 4.6). Questa definizione è corretta ma poco utile per il calcolo procedurale. Dimostriamo quindi un teorema utile.

Innanzitutto, assumiamo che in generale, per un problema LP $\mathcal{P}$ con n variabili decisionali e m vincoli, si ha che $n < m$. A questo punto, possiamo dire:

Definizione 5.1: Soluzione di base primale

Sia dato un problema LP $\mathcal{P}$ in forma primale standard con $n < m$. Sia $B \subseteq \{1, ..., m\}$ un sottoinsieme di indici di riga tale che $\text{card}(B) = n$. A questo punto, sia A_B la sottomatrice di A con con righe indicate da B, e b_B il sottovettore colonna di B con righe indicate da B, con det $\det A_B \neq 0$. Allora la soluzione di:

$$A_B x = b_B$$

è detta soluzione di base primale di $\mathcal{P}$.

da qui possiamo dimostrare il teorema:

Teorema 5.1: Caratterizzazione dei vertici primali

Su un problema in forma primale standard, un punto x del poliedro P è un vertice di P se e solo se è una soluzione di base primale ammissibile, ovvero:

$$x \in \text{vert}(P) \Leftrightarrow x \text{ è soluzione di base primale}$$

Riflettiamo un'attimo su questo risultato: se la definizione di vertice era effettivamente sufficiente a dichiarare *quali* erano i vertici, un teorema di caratterizzazione come quello sopra riportato ci fornisce una procedura per calcolarli tutti a partire da P. In questo, definizione e teorema di caratterizzazione sono effettivamente intercambiabili, ovvero:

$$\text{definizione} \Leftrightarrow \text{teorema di caratterizzazione}$$

6 Lezione del 01-10-24

6.1 Soluzioni di base primali degeneri

Abbiamo dato un teorema di caratterizzazione dei vertici primali. Questo teorema si basava sulla nozione di **soluzione di base primale**. Possiamo fare una distinzione fra soluzione di base degeneri e non degeneri:

Definizione 6.1: Soluzione di base degenere

Quando una soluzione di base è soluzione di più combinazioni delle disequazioni del problema, essa si dice degenere.

Questa definizione è esatta ma non particolarmente utile. Sostanzialmente, ci dice soltanto che una soluzione degenere è **ridondante** su più combinazioni di disequazioni (cioè risolve $A_B x = b_B$ su più permutazioni degli $1, ..., m$ elementi in classi n in B). Si noti che ridondante non significa **eliminabile**: questa affermazione purtroppo è vera soltanto in R^2, dove effettivamente si può rimuovere una delle disequazioni ridondanti ed avere sempre lo stesso risultato.

Diamo quindi una caratterizzazione delle soluzioni di base primali degeneri appoggiandoci al teorema di caratterizzazione dei vertici, ergo al concetto di soluzione di base:

Teorema 6.1: Caratterizzazione delle soluzioni di base primali degeneri

Se una soluzione è di base, ergo scelto $B = \{1, ..., m\}$ con $\text{card}(B) = n$ è data da $A_B x = b_B$, possiamo dire che è pure degenere quando $\exists i \in N$ t.c. $A_i x = b_i$, con $I = \{1, ..., m\} - B$.

Quindi, una soluzione di base è degenere quando almeno una variabile di base si annulla per almeno una delle disequazioni non di base indicate dagli indici I, che sono tutti gli indici fra $\{1, ..., m\}$ non contenuti in B.

Sulla stessa linea di pensiero, possiamo dimostrare un'altro teorema, stavolta sul concetto piuttosto intuitivo di ammissibilità. Potremmo infatti dire che una soluzione di base ammissibile, cioè che rientra all'interno della regione ammissibile, è tale se:

Teorema 6.2: Caratterizzazione delle soluzioni di base primali ammissibili

Se una soluzione è di base, ergo scelto $B = \{1, ..., m\}$ con $\text{card}(B) = n$ è data da $A_B x = b_B$, possiamo dire che è ammissible quando $\forall i \in N$ si ha $A_i x \leq b_i$, con $I = \{1, ..., m\} - B$.

cioè banalmente rispetta tutte le disequazioni.

6.1.1 Considerazioni numeriche sui numeri di soluzioni base

Solitamente un problema con n variabili decisionali a $m \geq n$ vincoli. Posti questi vincoli, visto che per calcolare $\text{vert}(P)$ prendiamo effettivamente tutte le combinazioni degli m vincoli classe n variabili decisionali, possiamo usare il coefficiente binomiale per calcolare il numero massimo di potenziali vertici:

$$\text{card}(\text{vert}(P)) \sim \binom{m}{n} = \frac{m!}{n!(m-n)!}$$

In verità, i vertici sono solitamente meno, in quanto possiamo rimuovere le soluzioni non ammissibili. Inoltre, le soluzioni degeneri non contribuiscono al risultato, ergo anche quelle non sono rilevanti.

6.2 Riassunto delle trasformazioni equivalenti

Riassumiamo adesso le trasformazioni equivalenti che abbiamo individuato finora per le disequazioni di problemi LP:

- $\min(C^\intercal \cdot x) \leftrightarrow \max(C^\intercal \cdot x)$: trasformiamo problemi di massimo in problemi di minimo invertendo i segni;
- $Ax \geq b \leftrightarrow -Ax \leq -b$: invertiamo il verso della diseguaglianza moltiplicando per -1;
- $Ax = b \rightarrow Ax \leq b \wedge Ax \geq b$: convertiamo un'uguaglianza in una coppia di diseguaglianze;
- $Ax \leq b \rightarrow Ax + s = b$: convertiamo una diseguaglianza in un'uguaglianza introducendo una variabile di surplus. Si nota che la variabile di surplus può essere riconosciuta per essere rimossa, da:
  - $s > 0$;
  - Compare in un solo vincolo, che è di uguaglianza;
  - Ha coefficiente 0 nella funzione obiettivo, e 1 nell'equazione dove compare;
- $x \gtrless 0 \rightarrow x = x^+ - x^-, \quad x^+ \geq 0, \quad x^- \geq 0$: aggiriamo il vincolo di positività introducendo parti positive e negative delle variabili decisionali, con rispettivi vincoli di positività.

Usiamo queste trasformazioni per portare i problemi LP in forme standard. Esistono molteplici forme standard, ma in questo corso ci riguardano: il formato *linprog*, usato dal pacchetto software *MATLAB*, le forme standard primale (già vista) e duale (che vedremo fra poco).

7 Lezione del 02-10-24

7.1 Trasporto

Poniamo il seguente problema:

Problema 7.1: Trasporto

Due centrali del latte di Firenze producono rispettivamente 50 e 60 mila litri di latte al giorno. Le centrali servono tre quartieri, che consumano rispettivamente 30, 30 e 20 mila litri di latte al giorno. Si conosce il costo necessario per portare un migliaio di litri di latte da ogni centrale a ogni quartiere, riportato nella seguente tabella:

	Novoli	**Statuto**	**Rifredi**
Centrale A	6	8	4
Centrale B	7	3	9

Vogliamo capire quanto latte deve spedire ogni centrale ad ogni quartiere.

Nota simpatica: secondo l'indagine INRAN-SCAI 2005-06, l'italiano medio consuma 0.115g di latte al giorno, che per un peso specifico di circa 1.040kg/L fanno 0.11L di latte al giorno. Al 2024, il comune di Firenze ha 364 073 abitanti, ergo dovrebbe avere bisogno di approssimativamente $40\ 258\text{L}$ di latte al giorno. I fiorentini nell'esempio devono avere le ossa veramente forti!

Possiamo esprimere il problema dell'esempio come un problema LP. Abbiamo innanzitutto che i costi di trasporto formano una matrice:

$$C_{matr} = \begin{pmatrix} 6 & 8 & 4 \\ 7 & 3 & 9 \end{pmatrix}$$

che possiamo linearizzare, come avevamo fatto nei problemi di assegnamento di costo minimo, in un vettore costo:

$$C = (6, 8, 4, 7, 3, 9)$$

Questo vettore moltiplica il vettore delle variabili decisionali, che è la linearizzazione della matrice:

$$x_{matr} = \begin{pmatrix} x_{13} & x_{14} & x_{15} \\ x_{23} & x_{24} & x_{25} \end{pmatrix}$$

Questa matrice non rappresenta altro che quanto latte mandare ad ogni quartiere.

A questo punto, possiamo stabilire i vincoli. Innanzitutto, non si può avere più latte di quanto viene prodotto, ergo:

$$\begin{cases} x_{13} + x_{14} + x_{15} \leq 50 \\ x_{23} + x_{14} + x_{15} \leq 60 \end{cases}$$

inoltre, si vuole fornire ad ogni quartiere il fabbisogno richiesto, ergo:

$$\begin{cases} x_{13} + x_{23} \geq 30 \\ x_{14} + x_{24} \geq 30 \\ x_{15} + x_{25} \geq 20 \end{cases}$$

Questo è un problema di programmazione lineare.

In generale, quindi, un problema di trasporto minimizza la funzione obiettivo data da una matrice di costo in $n \times m$ variabili, con m vincoli di riga sul vettore o_j dei limiti di produzione, e n vincoli di colonna sul vettore d_j della domanda, in forma:

$$\begin{cases} \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} = C^{\intercal} \cdot x \\ \sum_{i=1}^{m} x_{ij} \geq d_j \quad \forall j = 1, ..., n \\ \sum_{j=1}^{n} x_{ij} \leq o_i \quad \forall i = 1, ..., m \\ x \geq 0 \end{cases}$$

Non ci sono soluzioni se la domanda supera l'offerta, cioè sé:

$$\sum_{j=1}^{m} d_j \geq \sum_{i=1}^{m} o_i$$

Mentre in caso di eccessi di produzione, potremmo trasformare le diseguaglianze in uguaglianze, e aggiugnere un carico "fittizio" con costo zero dove deviare il surplus di produzione.

Inoltre, come avevamo detto per i problemi di assegnamento di costo minimo, anche qui potremmo scegliere di distinguere fra trasporti divisibili (nello spazio $x = \mathbb{R}^n$) e indivisibili (col vincolo aggiunto $x = \mathbb{Z}^n$).

7.2 Forma duale standard

Avevamo definito la forma primale standard:

$$\begin{cases} \max C^\intercal \cdot x \\ Ax \leq b \end{cases}$$

Introduciamo adesso la forma duale standard:

Definizione 7.1: Forma duale standard

Un problema di programmazione lineare si dice in forma duale standard quando è espresso in forma:

$$\begin{cases} \min(c^\intercal \cdot x) \\ Ax = b \\ x \geq 0 \end{cases}$$

7.2.1 Vertici del duale

Sulle forme duali è semplice il calcolo dei vertici. Possiamo infatti avere, come avevamo fatto sulla primale:

Definizione 7.2: Soluzione di base duale

Sia dato un problema LP $\mathcal{P}$ in forma duale standard. Sia $B \subseteq \{1, ..., n\}$ un sottoinsieme di indici di variabili decisionali tale che $\text{card}(B) = m$. Chiamiamo x_B l'insieme delle variabili decisionali individuate da B, e x_N l'insieme delle $n - m$ variabili decisionali rimanenti:

$$x = \{x_B, x_N\}$$

Impostiamo quindi tutte le x_N a 0: avremo un sistema di m variabili in m equazioni, quindi determinato. La soluzione di quel sistema è detta soluzione di base duale di $\mathcal{P}$.

Indichiamo spesso questo vertice come $(bA_B^{-1}, 0)$. Questa definizione porta ad una caratterizzazione dei vertici del tutto analoga a quella dichiarata sui problemi in forma primale standard:

Teorema 7.1: Caratterizzazione dei vertici duali

Su un problema in forma duale standard, un punto x del poliedro P è un vertice di P se e solo se è una soluzione di base duale ammissibile, ovvero:

$$x \in \text{vert}(P) \Leftrightarrow x \text{ è soluzione di base duale}$$

7.2.2 Soluzioni di base duali degeneri

Possiamo ricavare il concetto di soluzione degenere (e anche di soluzione ammissibile) sui vertici del poliedro del duale. Si ha:

Teorema 7.2: Caratterizzazione delle soluzioni di base duali degeneri

Se una soluzione è di base, ergo scelto $B = \{1, ..., n\}$ con $\text{card}(B) = m$ è data da $(bA_B^{-1}, 0)$, possiamo dire che è pure degenere quando $\exists i \in B$ tale che almeno una componente si annulla.

e riguardo l'ammissibilità:

Teorema 7.3: Caratterizzazione delle soluzioni di base duali ammissibili

Se una soluzione è di base, ergo scelto $B = \{1, ..., n\}$ con $\text{card}(B) = m$ è data da $(bA_B^{-1}, 0)$, possiamo dire che è ammissibile quando il vettore soluzione è ≥ 0.

8 Lezione del 03-10-24

8.1 Teoria della dualità

Introduciamo adesso uno dei concetti più importanti della programmazione lineare. Avevamo posto problemi LP in forma primale standard come:

$$\begin{cases} \min(c^\intercal \cdot x) \\ Ax \leq b \end{cases}$$

Ottimizzare questo problema significa partire dal basso e avvicinarsi verso un punto di massimo. Potremmo scegliere di seguire il percorso opposto: cercare di estrapolare un limite superiore per la soluzione dai vincoli, e minimizzarlo.

Per fare ciò introduciamo m variabili, una per ogni disequazione, che denoteremo come $y_1, ..., y_m$. Moltiplichiamo ogni disequazione per la y_i corrispondente a destra e a sinistra. Su un semplice problema $n, m = 2$, questo darà una forma del tipo:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 \leq b_1 \\ a_{21}x_1 + a_{22}x_2 \leq b_2 \end{cases} \rightarrow \begin{cases} y_1 \cdot (a_{11}x_1 + a_{12}x_2) \leq b_1 y_1 \\ y_2 \cdot (a_{21}x_1 + a_{22}x_2) \leq b_2 y_2 \end{cases}$$

Se vincoliamo gli y_i in modo che ogni variabile decisionale x_i del sistema abbia un coefficiente del costo $\geq c_i$ corrispondente, otterremo una disequazione che ha a sinistra una situazione di valore uguale o addirittura migliore di quella data dalla funzione costo, e a destra un massimo (che era ciò che stavamo cercando, un limite superiore).

Abbiamo quindi una serie di variabili vincolate:

$$\begin{cases} y_1 a_1 1 + y_2 a_2 1 \geq c_1 \\ y_1 a_1 2 + y_2 a_2 2 \geq c_2 \end{cases}$$

e una funzione da minimizzare:

$$\min(b_1 y_1 + b_2 y)$$

Cioè, ci siamo ricondotti ad un altro problema LP. Possiamo formalizzare questo risultato:

Definizione 8.1: Duale di un problema LP

Per un qualsiasi problema LP $\mathcal{P}$, detto primale, con $m \geq n$, possiamo definire il duale $\mathcal{D}$:

$$P : \begin{cases} \max(c^\intercal \cdot x) \\ Ax \leq b \end{cases} \quad \to D : \begin{cases} \min(b^\intercal \cdot y) \\ A^\intercal y = c \\ y \geq 0 \end{cases}$$

dove si nota $x \in \mathbb{R}^n$ e $y \in \mathbb{R}^m$.

Il duale viene posto in forma duale standard in quanto ciò che ci interessa è *stringere* il limite superiore fino al suo minimo, in un modo che fa combaciare perfettamente le variabili con il loro vettore costo, da cui le uguaglianze.

Si può dimostrare che l'operazione del calcolo del duale è involutoria: il duale del duale è nuovamente il primale, e così via.

8.1.1 Dualità debole

Visto che abbiamo costruito il duale per avere un limite superiore dei valori ottenuti dalla funzione obiettivo del primale, potremo dimostrare facilmente:

Teorema 8.1: Dualità debole

Se i poliedri P e il suo duale D non sono vuoti, allora:

$$c^\intercal x \leq y^\intercal b \quad \forall x \in P, \forall y \in D$$

Cioè il duale è sempre maggiore del primale.

8.1.2 Dualità forte

Idealmente, ciò che vorremmo è che primale e duale convergessero verso un punto comune, ergo l'ottimo di entrambi. Effettivamente, questo risultato è verificato:

Teorema 8.2: Dualità forte

Se i poliedri P e il suo duale D non sono vuoti, allora:

$$-\infty \leq \min_{y \in D} b^\intercal y = \max_{x \in P} c^\intercal x \leq +\infty$$

Il teorema della dualità forte afferma che, se entrambi i poliedri (primale e duale) sono non vuoti, allora condividono l'ottimo, e anzi, che due soluzioni nel primale e nel duale sono ottime solo se hanno lo stesso valore. Se invece solo il primale (solo il duale) è vuoto, si ha che entrambi condividono ottimo $-\infty$ (∞). Quando entrambi sono vuoti non si ha soluzione condivisa.

Si noti che la dualità è univocamente determinata solo nel caso di soluzioni non degeneri. Nel caso di soluzioni degeneri si può avere che a una soluzione del primale ne corrispondono multiple del duale, e viceversa. In questo caso, è possibile che si raggiunga un massimo (o un minimo) nel primale (nel duale), a partire da più vertici complementari, o addirittura senza che la soluzione di base complementare sia ammissibile. Più formalmente, se l'esistenza di una soluzione complementare ammissibile è **condizione necessaria e sufficiente** per problemi con soluzioni non degeneri, è solo **condizione sufficiente** per problemi con soluzioni degeneri.

Ad esempio, si noti il problema duale:

$$\begin{cases} \min 3y_1 - 7y_2 + 5y_3 + 22y_4 + 14y_5 + 15y_6 \\ -y_1 - y_2 + 3y_4 + 2y_5 + 2y_6 = 2 \\ y_1 - 4y_2 + y_3 + 2y_4 + y_5 - 2y_6 = 1 \\ ey_i \geq 0 \end{cases}$$

con il primale associato:

$$\begin{cases} \max 2x_1 + x_2 \\ -x_1 + x_1 \leq 3 \\ -x_1 - 4x_2 \leq -7 \\ x_2 \leq 5 \\ 3x_1 + 2x_2 \leq 22 \\ 2x_1 + x_2 \leq 14 \\ 2x_1 - 2x_2 \leq 5 \end{cases}$$

Con la base $B = \{1, 5\}$, si hanno le soluzioni $\bar{x} = (4, 5)$ e $\bar{y} = (0, 0, 0, 0, 1, 0)$. Si dimostra che $\bar{y}$ è ottima a D, mentre $\bar{x}$ è non ammissibile su P. Ciò deriva dal fatto che $\bar{y}$ in D è degenere: descrive un vincolo perpendicolare al vettore costo (di indice $i = 5$), e quindi soddisfatto da infinite soluzioni del primale. Il punto $\bar{x}$ nel primale ha come complementare l'ottimo del duale perchè sta proprio su questo vincolo.

8.1.3 Scarti complementari

Si può dimostrare il seguente teorema:

Teorema 8.3: Scarti complementari

Se le soluzioni x e y dei problemi primale e duale $\mathcal{P}$ e $\mathcal{D}$ sono entrambe ottime, allora vale:

$$y^\intercal(b - Ax) = 0$$

Questo si ricava dal fatto che, per la dualità forte, si ha che:

$$c^\intercal x = y^\intercal Ax = y^\intercal b \Rightarrow y^\intercal(b - Ax) = 0$$

Il significato del teorema è che, se una disequazione nel primale è *stretta*, allora la corrispondente variabile nel duale è $\neq 0$, e viceversa.

8.1.4 Soluzioni di base

Avevamo dato una definizione di soluzione di base per problemi LP in forma sia primale che duale. Possiamo dimostrare che non solo questa nozione esiste su entrambe le formule, ma è analoga su coppie primale / duale.

Avevamo posto che la formazione di una certa base $B \in \{1, ..., m\}$ per ricavare soluzioni di base. Per il primale, questo significa partizionare la matrice e i termini noti:

$$A = \left(\frac{A_B}{A_N}\right), \quad b = \left(\frac{b_A}{b_N}\right)$$

mentre per il duale, significherà partizionare le variabili introdotte:

$$y = \left(\frac{y_B}{y_N}\right)$$

noto il numero di $y_1, ..., y_m$ uguale a m.

Questo significa che possiamo trovare due soluzioni di base corrispondenti per un'unica base su primale e duale. Queste sono:

- Soluzione di base primale: $x = A_B^{-1} b_B$;
- Soluzione di base duale: $y_B^\intercal = c^\intercal A_B^- 1, \quad y_N = 0$;

Si dice che le soluzioni di base sono **complementari**.

Dimostrazione Vogliamo che $y^\intercal(b - Ax)$ sia $= 0$ soddisfatte le condizioni di base. Applichiamo quindi la base:

$$y^\intercal(b - Ax) = (y_B^\intercal, y_N^\intercal) \begin{pmatrix} b_B - A_B x \\ b_N - A_N x \end{pmatrix} = \left(c^\intercal A_B^{-1}, 0\right) \begin{pmatrix} b_B - A_B A_B^{-1} b_B \\ b_N - A_N A_B^{-1} b_B \end{pmatrix}$$

$$= \left(c^\intercal A_B^{-1}, 0\right) \begin{pmatrix} 0 \\ b_N - A_N A_B^{-1} b_B \end{pmatrix} = 0$$

Questo nome non è a caso, in quanto si può dimostrare le due soluzioni sono in scarti complementari. Da questo risultato, si ha che se entrambe le soluzioni sono ammissibili, cioé:

- La primale è ammissibile:
$$\forall i \in N \text{ si ha } A_i x \leq b_i$$
ergo i vincoli sono soddisfatti;
- La duale è ammissibile:
$$y \geq 0$$

questo è condizione sufficiente perche la soluzione sia ottima, e dagli scorsi corollari, sia l'ottima sia del primale che del duale.

Formalizziamo quanto detto in un teorema:

Teorema 8.4: Condizioni di ottimalità di soluzione di base

Dato un vertice del primale, ottenuto da una certa base, si può costruire il complemento duale sulla stessa base. Se entrambi i vertici ottenuti sono ammissibili, allora sono uguali e ottimi dei rispettivi problemi.

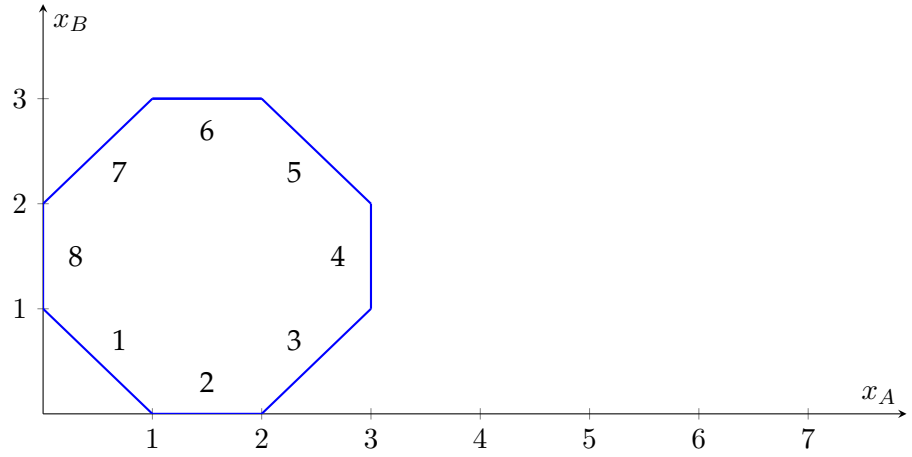
9 Lezione del 07-10-24

9.1 Algoritmo del simplesso primale

Supponiamo di avere un problema LP in formato primale standard con $n = 8$ vincoli, espresso come:

$$\begin{cases} \max(c^\intercal \cdot x) \\ Ax \leq b \end{cases}$$

e con poliedro:



Scegliamo un vertice di partenza, per adesso ad arbitrio: diciamo $\bar{x} = (0, 1)$ (vedremo in seguito un'algoritmo particolare per ricavare un vertice, che ci permetterà anche di determinare se il poliedro è vuoto o meno). Ci chiediamo se questo vertice $\bar{x}$ è ottimo. Visto che è vertice, abbiamo che per una matrice A_B e un vettore b_B di base:

$$\bar{x} = A_B^{-1} b_B$$

e che possiamo costruire il complementare duale $\bar{y}$, impostando a zero le variabili fuori base e risolvendo il sistema:

$$\bar{y} = (cA_B^{-1}, 0)$$

e applicare il test di ottimalità, cioè vedere se:

$$cA_B^{-1} \geq 0$$

ergo $\bar{y} \in D$, quindi il complementare duale esiste e il vertice è ottimo. Se questa condizione risulta verificata, possiamo fermarci, in quanto abbiamo trovato la soluzione ottimale.

In caso contrario, avremo $\exists k \in B$ tale che $\bar{y}_k < 0$. Dovremo quindi spostarci verso un'altro vertice, magari *adiacente*, che dal punto di vista delle basi, significa cambiare un solo indice di base, conservando gli altri. Possiamo formalizzare questa affermazione definendo un **indice uscente** h ed un **indice entrante** k. Sostituire un indice di base significa effettuare il cambio di base:

$$B := B \setminus \{h\} \cup \{k\}$$

Resta la domanda di *quale* spigolo scegliere: in uno spazio vettoriale $\mathbb{R}^n$, ho a disposizione n spigoli che si staccano dallo stesso vertice. Ovviamente, vorrei scegliere uno spigolo che accresce la funzione obiettivo, e si può dimostrare che ne esiste almeno uno: altrimenti sarei già all'ottimo. Inoltre, avendo un metodo per scegliere sempre lo spigolo di crescita maggiore potrei dire 2 cose: l'algoritmo tende all'ottimo (il vertice da cui non si staccano spigoli che accrescono la funzione obiettivo), e termina in un numero finito di passi (prima o poi raggiungerà inevitabilmente un vertice che massimizza la funzione).

Prima però dobbiamo chiarire una questione: scegliere un nuovo spigolo significa trovare 2 indici base, uno da eliminare e uno da inserire. Si può dire che il primo indice, quello uscente, indica anche la direzione di spostamento: allentando un vincolo ci spostiamo sulla semiretta del prossimo. Allo stesso tempo, scegliere un indice da rimuovere non basta: dobbiamo scegliere quale introdurre, che geometricamente significa capire *quanto* ci possiamo spostare lungo la semiretta prima di uscire dalla regione di ammissibilità. Vediamo quindi questi due passaggi in ordine.

- **Indice uscente**
  Diciamo:
  $$W = \left(-A_B^{-1}\right)$$
  e prendiamo le colonne W^i corrispondenti agli indici di base scelti.

  Possiamo allora dire che l'equazione degli spigoli dati dalle disequazioni all'indice i sono:
  $$\bar{x} + \lambda W^i$$

  Mettiamo questa equazione nella funzione costo:
  $$c\left(\bar{x} + \lambda W^i\right) = c\bar{x} + \lambda c W^i$$

  Qui abbiamo $c\bar{x}$, che è il valore nel vertice, e un'altro termine scalato da λ. Ricordiamo poi che $cA_B^{-1} = \bar{y}_B$, e che $W = \left(-A_B^{-1}\right)$, ergo $cW^i = -\bar{y}_B$:
  $$c\bar{x} + \lambda c W^i = c\bar{x} - \lambda \bar{y}_B$$

  Vogliamo quindi "allentare" l'indice (e il corrispettivo vertice) che ci dà $\bar{y}_i < 0$, in quanto è quello che restituisce un $cW^i > 0$, e quindi un accrescimento della funzione. Definiamo allora questo indice:

  Definizione 9.1: Indice uscente primale

  Chiamiamo indice uscente h, da una certa soluzione della base B:
  $$h := \min\{i \in B \text{ t.c. } \bar{y}_i < 0\}$$

  Il $\min$ significa che in caso di più i negativi, si adotta la regola anticiclo (di Bland) di scegliere il primo. In caso di nessun i negativo, la complementare duale esiste e siamo sull'ottimo.

- **Indice entrante**
  Adesso cerchiamo per quali λ lo spigolo $\bar{x} + \lambda W^h$ resta ammissibile, ergo soddisfa:
  $$A_i\left(\bar{x} + \lambda W^h\right) \leq b_i, \quad i \in N$$

Questo significa effettivamente vedere qual'è il primo vincolo che "stringiamo", o che incontriamo, spostandoci lungo la semiretta ottenuta allentando il vincolo dato dall'indice uscente.

Possiamo dire:

$$A_i\left(\bar{x} + \lambda W^h\right) = A_i\bar{x} + \lambda A_i W^h \leq b_i$$

da cui si ricava (e si risolve) la disequazione di primo grado:

$$\lambda A_i W^h \leq b_i - A_i\bar{x} \Rightarrow \lambda \leq \frac{b_i - A_i\bar{x}}{A_i W^h}$$

Notiamo che se fosse $A_i W^h \leq 0,\ \forall i \in N$, avremmo che l'indice rappresenta una direzione di regressione, in quanto $\lambda \to +\infty$. Si ha quindi che il duale non ha soluzione, e il primale $\to +\infty$. In caso contrario, noi vogliamo trovare il primo vincolo che si va a stringere, quindi dovremo calcolare tutti gli r_i:

$$r_i = \frac{b_i - A_i\bar{x}}{A_i W^h}, \quad i \in N, \quad A_i W^h > 0$$

e scegliere l'indice che dà $\vartheta = \min(r_i)$. Definiamo allora anche questo indice:

Definizione 9.2: Indice entrante primale

Chiamiamo indice entrante k, da una certa soluzione della base B e un certo indice uscente h:

$$k := \min\{i \in N \text{ t.c. } A_i W^h > 0, \quad \frac{b_i - A_i\bar{x}}{A_i W^h} = \vartheta\}$$

Anche qui, il $\min$ serve a selezionare il primo indice valido, ed è una regola anticiclo (di Bland). Notiamo due possibili situazioni:

- Si potrebbero avere più r_i uguali: questi rappresentano soluzioni di base degenere *in arrivo*, in quanto sono più modi di arrivare allo stesso vertice stringendo vincoli diversi;
- Si potrebbe avere un r_i nullo: questo significa che il vertice è sullo stesso vertice da dove siamo partiti, ergo rappresenta una soluzione di base degenere *in partenza*.

Come prima, le regole anticiclo di Bland assicurano anche che l'algoritmo non si blocchi a ciclare su queste soluzioni degeneri.

Abbiamo quindi tutti gli strumenti necessari alla formulazione dell'algoritmo del simplesso:

Algoritmo 1 del simplesso primale

Input: un problema LP in forma primale standard
Output: la soluzione ottima
Trova una base B che genera una soluzione di base primale ammissibile
ciclo:
Calcola la soluzione di base primale $\bar{x} = A_b^{-1} b_B$ e la soluzione di base duale $\bar{y} = (cA_b^{-1}, 0)$
if $\bar{y_B} \geq 0$ **then**
Fermati, $\bar{x}$ è ottima per P e $\bar{y}$ è ottima per D
else
Calcola l'indice uscente:

$$h := \min\{i \in B \text{ t.c. } \bar{y_i} < 0\}$$

poni $W := -A_B^{-1}$ e indica con W^h la h-esima colonna di W
end if
if $A_i W^h \leq 0 \quad \forall i \in N$ **then**
Fermati, $P \to +\infty$ e D non ha soluzione ottima
else
Calcola:

$$\vartheta = \min\{\frac{b_i - A_i \bar{x}}{A_i W^h} \text{t.c.} \quad i \in N, \quad A_i W^h > 0\}$$

e trova l'indice entrante:

$$k := \min\{i \in N \text{ t.c. } A_i W^h > 0, \quad \frac{b_i - A_i \bar{x}}{A_i W^h} = \vartheta\}$$

end if
Aggiorna la base come:

$$B := B \setminus \{h\} \cup \{k\}$$

Torna a ciclo

10 Lezione del 08-10-24

Abbiamo trovato un'algoritmo per ricavare una soluzione ottimale partendo da una soluzione di base primale ammissibile. Nel fare ciò, si è dato per scontato che l'algoritmo di partenza ci avrebbe fornito una soluzione di base primale ammissibile, e che qualsiasi successivo passo del simplesso ci avrebbe restituito altre soluzioni di base primali ammissibili. In verita, le soluzioni di base trovate possono avere le seguenti combinazioni di ammissibilità su P primale e D duale:

- **Ammissibile su P e D:** in questo caso siamo all'ottimo dalla dualità forte;
- **Ammissibile su P ma non su D:** in questo caso siamo su un comune punto ammissibile non ottimo;
- **Non ammissibile su P o D:** in questo caso si scarta la soluzione;
- **Ammissibile su D ma non su P:** effettivamente, ancora non abbiamo un modo per gestire questa situazione.

Per questo motivo estendiamo l'algoritmo del simplesso al duale.

10.1 Algoritmo del simplesso duale

Intendiamo mantenere, ad ogni passo, la soluzione di base duale come ammissibile e controllare l'ammissibilità di quella primale. Se la soluzione di base primale è ammissibile (l'inverso di come avevamo visto per il simplesso primale), allora siamo sull'ottimo. Altrimenti, si cambia base, cercando di minimizzare la funzione obiettivo su una nuova soluzione di base duale.

Abbiamo quindi un vertice del poliedro duale, cioè una soluzione di base ammissibile del duale. Ci chiediamo se questo vertice $\bar{y}$ è ottimo. Visto che è vertice del duale, abbiamo che per una matrice A_B e un vettore costo c del primale:

$$\bar{y} = (cA_B^{-1}, 0), \quad \text{con } cA_B^{-1} \geq 0$$

dove ricordiamo questa notazione significa impostare le variabili non di base $\in N$ a 0 e risolvere il sistema in m variabili rimasto.

Adesso possiamo costruire il complementare primale $\bar{x}$, ponendo:

$$\bar{x} = A_B^{-1} b_b$$

e applicare il test di ammissibilità, cioè vedere se:

$$A_N(A_B^{-1} b_B) \leq b_N$$

ergo $\bar{x} \in P$, quindi il complementare duale esiste e il vertice è ottimo. Come prima, se questa condizione risulta verificata possiamo fermarci, in quanto abbiamo trovato la soluzione ottimale.

In caso contrario, avremo $\exists i \in N$ tale che $b_i - A_i(A_B^{-1} - b_B) < 0$. Questo equivale a ciò che avevamo trovato per il primale per quanto riguardava gli indici uscenti, con una sola differenza: per risolvere il duale, si trova **prima** l'**indice entrante**, e **poi** l'**indice uscente**. Vediamo quindi i passaggi in ordine:

- **Indice entrante**
  Abbiamo che, se il vertice non è ottimo, vale:

  $$\exists i \in N \text{ t.c. } b_i - A_i(A_B^{-1} - b_B) < 0$$

  Vogliamo che l'indice entrante sia quell che raggiunge questo valore negativo, quindi:

  Definizione 10.1: Indice entrante duale

  Chiamiamo indice entrante k, da una certa soluzione della base B:

  $$k := \min\{i \in N \text{ t.c. } b_i - A_i\bar{x} < 0\}$$

  Come sempre, $\min$ significa che in caso di più i negativi, si adotta la regola anticiclo (di Bland) di scegliere il primo. In caso di nessun i negativo, la complementare duale esiste e siamo sull'ottimo.

- **Indice uscente**
  Cerchiamo quindi l'indice uscente. Dovremo prima definire la matrice W come $W = -A_B^{-1}$, e calcolare il prodotto (perlopiù analogo al primale, ma si noti il segno della diseguaglianza capovolto):

  $$A_k W^i < 0$$

  Questo ci fornisce una regola, come nel simplesso primale, per l'esistenza di una soluzione: nel caso non sia verificato, si ha che il duale $\to -\infty$ e che il primale è vuoto. In caso contrario, si possono calcolare i rapporti, come:

  $$r_i = \frac{-\bar{y}_i}{A_k W^i}, \quad i \in B, A_k W^i < 0$$

  e scegliere l'indice che da $\vartheta = \min r_i$, cioè:

  Definizione 10.2: Indice uscente duale

  Chiamiamo indice uscente h, da una certa soluzione della base B e un certo indice entrante k:

  $$h := \min\{i \in B \text{ t.c. } A_k W^i < 0, \quad \frac{-\bar{y}_i}{A_k W^i} = \vartheta\}$$

  Anche qui, il $\min$ serve a selezionare il primo indice valido, ed è una regola anticiclo.

Abbiamo quindi revisionato tutti gli strumenti necessari alla formulazione dell'algoritmo del simplesso, stavolta duale:

Algoritmo 2 del simplesso duale

Input: un problema LP in forma primale standard
Output: la soluzione ottima
Trova una base B che genera una soluzione di base duale ammissibile.
ciclo:
Calcola la soluzione di base duale $\bar{y} = (cA_B^{-1}, 0)$ e la soluzione di base primale $\bar{x} = A_B^{-1} b_B$
if $A_N(A_B^{-1}b_B) \leq b_N$ **then**
Fermati, $\bar{y}$ è ottima per D e $\bar{x}$ è ottima per P
else
Calcola l'indice entrante:

$$k := \min\{i \in N \text{ t.c. } b_i - A_i\bar{x} < 0\}$$

poni $W := -A_B^{-1}$ e indica con W^i la i-esima colonna di W
end if
if $A_k W^i \geq 0 \quad \forall i \in B$ **then**
Fermati, $D \to -\infty$ e P non ha soluzione ottima
else
Calcola:

$$\vartheta = \min\{\frac{-\bar{y}_i}{A_k W^i} \text{t.c.} \quad i \in B, \quad A_k W^i < 0\}$$

e trova l'indice uscente:

$$h := \min\{i \in B \text{ t.c. } A_k W^i < 0, \quad \frac{-\bar{y}_i}{A_k W^i} = \vartheta\}$$

end if
Aggiorna la base come:

$$B := B \setminus \{h\} \cup \{k\}$$

Torna a ciclo

11 Lezione del 09-10-24

11.1 Problema duale ausiliario

Fino ad ora abbiamo rimandato la trattazione dell'algoritmo per determinare se un poliedro è vuoto. Diamo adesso quest'algoritmo, notando inoltre che, nel caso il poliedro non fosse vuoto, dovrebbe fornirci un possibile vertice di partenza per l'algoritmo del simplesso.

Partiamo da un poliedro in forma duale standard:

$$\begin{cases} Ax = b \\ x \geq 0 \end{cases}$$

dove m è il numero di variabili e n il numero di vincoli, e quindi $A : m \times n$.

Adottiamo questa forma in quanto siamo sicuri (a differenza della forma primale) che non presenti linealità. Costruiamo quindi quello che viene chiamato **duale ausilia-**

rio, DA:

$$\begin{cases} Ax = b \\ x \geq 0 \end{cases} \rightarrow \begin{cases} \min \sum_{i=1}^{n} \varepsilon_i \\ Ax + I\varepsilon = b \\ x_i \varepsilon \geq 0 \end{cases}$$

dove ε rappresenta un vettore di variabili ausiliarie di dimensione n, cioè una per ogni equazione. Possiamo rappresentare il sistema ottenuto anche attraverso due matrici a blocchi:

$$(A|I) \begin{pmatrix} x \\ \varepsilon \end{pmatrix}$$

Chiamiamo quindi $v(\text{DA})$ la soluzione del duale ausiliario, e formuliamo il teorema:

Teorema 11.1:

Se $v(\text{DA}) > 0$, allora $D = \emptyset$, altrimenti, se $v(\text{DA}) = 0$, $D \neq \emptyset$. Equivale a dire:

$$v(\text{DA}) = 0 \Leftrightarrow D \neq \emptyset$$

Dobbiamo però capire come risolvere il duale ausiliario, magari senza usare questo teorema, in quanto questo si andrebbe a creare una catena infinita di duali ausiliari da risolvere... Notiamo quindi che il duale ausiliario ha sempre una base plausibile, che è quella data dagli indici delle ultime n variabili, quelle introdotte come ε_i.

Non solo, abbiamo anche che:

Teorema 11.2:

La soluzione ottima di (DA), se $D \neq \emptyset$, ci fornisce un vertice di D stesso.

Notiamo che, visto che il primo teorema chiedeva $v(\text{DA}) = 0$ perché $D \neq \emptyset$, allora si ha che nella soluzione ottima di DA nulla (quella che dimostra la non vuotezza del poliedro), le variabili ε_i sono nulle.

Il procedimento sarà quindi:

1. Prendere il problema duale;
2. Ricavare il duale ausiliario inserendo nei vincoli il vettore di n variabili ausiliarie ε;
3. Risolvere il duale ausiliario attraverso il simplesso duale, prendendo come passo iniziale la base data dagli ultimi n indici, cioè che comprende le variabili ausiliarie appena introdotte;
4. Fare $\geq n$ passi al simplesso, aspettandoci che i primi n passi rimuovano tutte le variabili ausiliarie.

Il vertice che ricaviamo dall'algoritmo prende il nome di **soluzione ammissibile di base**.

11.2 Ricavare le variazioni dai cambi di base

Abbiamo che svolgere un passo al simplesso (primale o duale) significa effettuare un cambio di base, con base entrante e base uscente. Possiamo calcolare valori definiti, che

la soluzione di base sia ammissibile o meno, per la funzione obiettivo del primale e del duale per qualsiasi base.

Potremmo voler calcolare qual'è la variazione di valore di questa funzione dato un certo cambio di base. Ricordiamo quindi di aver introdotto la formula per spostarci lungo i vincoli di un certo λ sul primale:

$$cx(\lambda) = c\bar{x} - \lambda \bar{y}_h$$

e sul duale:

$$by(\lambda) = b\bar{y} + \lambda \left(b_h - A_k \bar{x}\right)$$

Queste due formule stabiliscono un legame fra il valore della funzione obiettivo e quello di determinate soluzioni $\bar{x}$ e $\bar{y}$ sottoposte a perturbazioni nella direzione dei vertici entranti (cioè allentando il vincolo h) di valore λ. Possiamo quindi usarle per calcolare quanto di chiedevamo all'inizio del paragrafo: la variazione del valore delle funzioni obiettivo del primale e del duale dopo il cambio di base, ricordando che il nostro λ sarà il rapporto r_i (in particolare lo avevamo chiamato ϑ) scelto per determinare il vincolo entrante (primale) o uscente (duale):

$$r_p = \frac{b_k - A_k \bar{x}}{A_k W^h}, \quad r_d = \frac{-\bar{y}_h}{A_k W^h}$$

11.3 Riassunto sulle regole anticiclo

Abbiamo introdotto, nel corso della trattazione degli algoritmi del simplesso, le **regole anticiclo di Bland**. Possiamo riassumerle come segue:

- **Simplesso primale**
  - **Regola anticiclo per l'indice uscente:** quando si calcola l'indice uscente h del simplesso primale, si considera una certa soluzione di base complementare duale $\bar{y} = \left(cA_b^- 1, 0\right)$. Se consideriamo ogni componente di $\bar{y}$ come il "prezzo ombra" associato ad ogni vincolo, cioè il guadagno (in verità l'opposto del guadagno) che potremmo avere rilassando quel vincolo, sembrerebbe conveniente prendere l'indice del componente più negativo come indice uscente. In verità, su certi problemi con soluzioni degeneri questo potrebbe causare cicli infiniti sulle basi degeneri. Per questo si adotta la regola anticiclo di prendere il primo indice negativo. Così si evitano cicli anche su basi degeneri, cioè ci si assicura che l'algoritmo termina in un numero finito di passi:

    $$h := \min\{i \in B \text{ t.c. } \bar{y}_i < 0\}$$

  - **Regola anticiclo per l'indice entrante:** allo stesso modo, quando si calcola l'indice entrante k del simplesso primale, si considerano i rapporti, che rappresentano le distanze sulla base definita dagli indici di base scelti degli altri vincoli, per scegliere il prossimo vincolo più vicino:

    $$r_i = \frac{b_i - A_i \bar{x}}{A_i W^h}, \quad \min(r_i) = \vartheta$$

    Si potrebbe avere che più indici hanno rapporto che corrisponde a ϑ. Anche in questo caso è necessario scegliere sempre l'indice più piccolo, in quanto questi rappresentano basi degeneri, ovvero un punto che resta vertice su diverse basi, e senza un'apposita regola si potrebbe finire per ciclare:

    $$k := \min\{i \in N \text{ t.c. } A_i W^h > 0, \quad \frac{b_i - A_i \bar{x}}{A_i W^h} = \vartheta\}$$

- **Simplesso duale**
  - **Regola anticiclo per l'indice duale:** nel simplesso duale, prima si calcola l'indice entrante k considerando la soluzione di base primale, ovvero prendendo gli i tali che $b_i - A_i\bar{x} < 0$. Potrebbero esistere più i che rispettano la proprietà, e bisogna quindi prendere l'indice i più piccolo:

    $$k := \min\{i \in N \text{ t.c. } b_i - A_i\bar{x} < 0\}$$

  - **Regola anticiclo per l'indice entrante:** infine, nel simplesso duale, si calcola l'indice uscente h prendendo i rapporti, analoghi a quelli del simplesso primale:

    $$r_i = \frac{\bar{y_i}}{A_k W^i}, \quad \min(r_i) = \vartheta$$

    anche qui si potrebbero avere più rapporti "distanza" con rapporto ϑ. Di questi va nuovamente preso quello con l'indice minore:

    $$h := \min\{i \in B \text{ t.c. } A_k W^i < 0, \quad \frac{-\bar{y_i}}{A_k W^i} = \vartheta\}$$

In generale, si può dire che in condizioni di sicurezza di non esistenza di basi degeneri, queste regole potrebbero non essere applicare. In tutti gli altri casi, evitano eventuali cicli sulle stesse basi degeneri.

11.4 Casi degeneri

Concludiamo infine la trattazione della PL notando il motivo dell'uso delle regole anticiclo di Bland nel calcolo degli indici uscenti ed entranti nell'algoritmo del simplesso. Prendiamo un problema di esempio:

$$\begin{cases} \max(x_1 + x_2) \\ x_1 \leq 1 \\ x_2 \leq 1 \\ x_1 + x_2 \leq 2 \\ x_i \geq 0 \end{cases}$$

Si ha che il vertice del poliedro $\bar{x} = (1, 1)$ è degenere: possiamo ottenerlo dalle basi $B = \{1, 2\}$, $B = \{1, 3\}$ e $B = \{2, 3\}$. In questo caso tutto funziona perché il vertice è anche ottimo, in caso contrario potremmo, se non si usassero le regole anticiclo di Bland, finire in un ciclo infinito.

Regole alternative a quella di Bland sono la scelta dell'indice sempre maggiore, o degli scarti o rapporti (apparentemente) ottimi, cioè più piccoli:

$$b_k - A_k\bar{x} = \min_{i \in N} (b_i - A_i\bar{x})$$

Queste regole, sopratutto l'ultima, potrebbero sembrare equivalenti o migliori di quelle di Bland. Invece è importante ricordare che potrebbero causare cicli.

12 Lezione del 10-10-24

12.1 Introduzione alla programmazione lineare intera

Iniziamo adesso a studiare un tipo di problemi che finora avevamo menzionato, ma mai risolto. Si pone il seguente:

Problema 12.1: Caricamento

Un ladro è riuscito a scassinare un'importante caveau, e ha di fronte a sé una serie di gemme preziose. Tenendo conto del peso e del tipo delle gemme, ricava una tabella col valore e il peso di ogni gemma:

Valore	2	4	7	9	13	16
Peso	6	8	9	11	15	18

Amesso che il suo zaino non possa portare più di 30 kg, quali gemme dovrà scegliere per avere il maggiore profitto possibile?

Questo problema riprende il celebre *knapsack problem*, che risulta essere NP-completo. Attraverso la **programmazione lineare intera**, possiamo ricavare soluzioni, almeno approssimate.

Si ha che qualsiasi possibilità delle 2^n configurazioni di gemme che il ladro può portare sono rappresentate da un vettore:

$$x = (x_0, ..., x_n), \quad x_i \in \{0, 1\}$$

e che i il valore e il peso delle stesse sono vettori:

$$v = (2, 4, 7, 9, 13, 16)$$

$$p = (6, 8, 9, 11, 15, 18)$$

Possiamo quindi formulare il sistema:

$$\begin{cases} \max\left(2x_1 + 4x_2 + 7x_3 + 9x_4 + 13x_5 + 16x_6\right) \\ 6x_1 + 8x_2 + 9x_3 + 11x_4 + 15x_5 + 18x_6 \leq 30 \\ x \in \{0, 1\}^n \end{cases} \rightarrow \begin{cases} \max\left(v^\intercal x\right) \\ p^\intercal x \leq P \\ x \in \{0, 1\}^n \end{cases}$$

dove P è il vincolo di massimo dei pesi.

Questo problema ricalca la struttura di un problema LP, ma il vincolo finale, che limita i valori possibili delle componenti del vettore soluzione, lo rende IPL. Decidiamo di trasformare il problema in un LP corrispondente, che chiameremo **rilassamento continuo**, e risolvere quello.

Definizione 12.1: Rilassato continuo

Dato un problema ILP in forma:

$$\begin{cases} \max c^\intercal \cdot x \\ Ax \leq b \\ x \in \mathbb{Z}^n \end{cases}$$

chiamiamo **rilassato continuo** il problema LP che rimuove il vincolo di interezza:

$$\begin{cases} \max c^\intercal \cdot x \\ Ax \leq b \end{cases}$$

Possiamo ricavare questo rilassato continuo in due modi: il primo prevede di rimuovere semplicemente il vincolo, magari includendo la positività, o ancor meglio l'appartenenza $x \in [0, 1]$:

$$\begin{cases} \max (v^\intercal x) \\ p^\intercal x \leq P \\ 0 \leq x \leq 1 \end{cases}$$

Questo metodo è sicuramente funzionante, ma preferiamo calcolare il duale, a partire dal primale con imposta la positività:

$$\begin{cases} \max (2x_1 + 4x_2 + 7x_3 + 9x_4 + 13x_5 + 16x_6) \\ 6x_1 + 8x_2 + 9x_3 + 11x_4 + 15x_5 + 18x_6 \leq 30 \\ -x_1 \leq 0 \\ -x_2 \leq 0 \\ -x_3 \leq 0 \\ -x_4 \leq 0 \\ -x_5 \leq 0 \\ -x_6 \leq 0 \end{cases} \rightarrow \begin{cases} \min (30y_1) \\ 6y_1 - y_2 = 2 \\ 8y_1 - y_3 = 4 \\ 9y_1 - y_4 = 7 \\ 11y_1 - y_5 = 9 \\ 15y_1 - y_6 = 13 \\ 18y_1 - y_7 = 16 \\ y_i \geq 0 \end{cases}$$

da cui notiamo che le disequazioni in forma $a_i x + s_i = b_i$ si possono rendere come $a_i x \geq b_i$:

$$\begin{cases} \min (30y_1) \\ 6y_1 \geq 2 \\ 8y_1 \geq 4 \\ 9y_1 \geq 7 \\ 11y_1 \geq 9 \\ 15y_1 \geq 13 \\ 18y_1 \geq 16 \\ y_1 \geq 0 \end{cases} \equiv \begin{cases} \min (30y_1) \\ y_1 \geq \frac{1}{3} \\ y_1 \geq \frac{1}{2} \\ y_1 \geq \frac{7}{9} \\ y_1 \geq \frac{9}{11} \\ y_1 \geq \frac{13}{15} \\ y_1 \geq \frac{8}{9} \\ y_1 \geq 0 \end{cases}$$

Abbiamo che questo sistema è banale: il minimo di $30y_1$ sottoposto ai vincoli si avrà quando y_1 rispetta i vincoli, che sono tutti $\geq$, e rispettare qualsiasi vincolo significa rispettare il vincolo con b_i più grande, quindi in questo caso $\frac{8}{9}$ Si ha quindi che la soluzione ottima del duale è $\frac{8}{9}$, e possiamo trovare con la stessa facilità la soluzione del primale:

abbiamo che la disequazione da cui si è ricavato la soluzione del duale ha indice $j = 6$. Si prende quindi la variabile con lo stesso indice nel primale, e si **satura**, cioè si porta al livello più alto possibile prima di violare il limite. Si nota che il valore ottimo trovato nel primale e nel duale (che è uguale dalla dualità forte) è $v = P\bar{y}_j = v_j\bar{x}_j$, con $\bar{x}$ e $\bar{y}$ le soluzioni ottime trovate rispettivamente al primale e al duale.

Nel nostro caso, $18x_6 \leq 30$, quindi $x_6 = \frac{30}{18}$, con valore $v = 16\frac{30}{18} = \frac{80}{3}$.

Questo approccio sarebbe quello che si applicherebbe intuitivamente senza sapere nulla di PL. Infatti i rapporti calcolati nel duale:

$$r_i = \frac{v_i}{p_i}$$

sono effettivamente i **rendimenti** di ogni elemento di peso e valore, cioè quanto valore portano in rapporto al loro peso. Chiaramente, vorremmo massimizzare gli elementi con rendimento maggiore, quindi saturiamo gli elementi con r_i massimo. Possiamo quindi definire il seguente algoritmo:

Algoritmo 3 caricamento per soluzioni ottime del rilassato continuo

Input: il rilassato continuo di un problema di caricamento
Output: la soluzione ottima $\bar{x}$
Considera i $r_i = \frac{v_i}{p_i}$ rendimenti di ogni elemento
Scegli l'indice j del rendimento massimo r_j
Satura la variabile j

Abbiamo che la soluzione trovata è un limite superiore per il problema di partenza, quello ILP: infatti, rilassando i vincoli, otteniamo una regione ammissibile più grande, e quindi aumentiamo il massimo. In verità anche l'arrotondamento in basso della soluzione v è un limite superiore: nel caso del problema abbiamo $\lfloor \frac{80}{3} \rfloor = 26$.

Un'idea banale per trovare il punto ottimo di questo problema ILP potrebbe essere quello di arrotondare il punto ottimo del rilassato continuo: nel nostro esempio, troviamo $\bar{x} = (0, 0, 0, 0, 0, 1)$, che dà massimo $v = 16$.

Notiamo un risultato fondamentale: se chiamiamo il valore arrotondato della soluzione del rilassato continuo v_s, e il valore ottenuto nell'approssimazione del *punto* di ottimo v_i, è vero che:

$$v_i \leq v_{ILP} \leq v_p$$

dove v_{ILP} è la soluzione del problema ILP che stiamo cercando. Per la precisione, possiamo definire l'errore:

$$\varepsilon = \frac{v_s - v_i}{v_i}$$

che sul problema in esame dà $\varepsilon \approx 60\%$, chiaramente poco utile. Sarà necessario adottare un qualche metodo per migliorare la stima data dall'arrotondamento del punto di ottimo del rilassato continuo.

Per adesso, generalizziamo quando trovato finora: bisogna distinguere fra v_i e v_s che si parli di problemi di minimizzazione o di massimizzazione. Si ha infatti che:

Teorema 12.1: Bound di problemi ILP

Per un dato problema ILP con soluzione v_{ILP}, si ha che:

$$v_i \leq v_{PLI} \leq v_s$$

- Se il problema è di **massimizzazione**, v_i è il valore calcolato soluzione ammissibile del rilassato continuo arrotondata **per difetto**, e v_s è la soluzione ottima del rilassato continuo arrotondata **per difetto**;
- Se il problema è di **minimizzazione**, v_i è la soluzione ottima del rilassato continuo arrotondata **per eccesso**, e v_i è il valore calcolato soluzione ammissibile del rilassato continuo arrotondata **per eccesso**;

12.1.1 Vincoli booleani

Possiamo chiamare il vincolo introdotto prima, $x \in \{0,1\}^n$, **vincolo booleano** (in modo più o meno informale). Come abbiamo detto, questo vincolo può essere reso come $x_i \geq 0$, o ancor meglio $0 \geq x_i \geq 1$ quando si porta al rilassato continuo (abbiamo usato il primo insieme di vincoli per ricavare il duale del rilassato continuo).

Introduciamo quindi l'algoritmo, equivalente a quello presentato prima per :

Algoritmo 4 caricamento per soluzioni ottime del rilassato continuo con vincolo booleano

Input: il rilassato continuo di un problema con vincolo booleano
Output: la soluzione ottima $\bar{x}$
Considera i $r_i = \frac{v_i}{p_i}$ rendimenti di ogni elemento
Scegli l'indice j del rendimento massimo r_j
Satura la variabile j
Quando finisci lo spazio, satura con il bene rimanente (sic.)

13 Lezione del 14-10-24

13.1 Algoritmi dei rendimenti

Riassumiamo i quattro algoritmi presentati finora per le valutazioni inferiori e superiori di problemi di ILP (con vincolo $x \in \mathbb{Z}^n$). Prendiamo in esempio lo "zaino": con $P = 7$. Il

v	10	17	22	21
p	4	5	6	2

vettore dei rendimenti sarà quindi:

$$r = \left(\frac{10}{4}, \frac{17}{5}, \frac{22}{6}, \frac{21}{2}\right) \approx (2.5, 3.4, 3.66, 10.5)$$

- **Problema booleano**

$$\begin{cases} \max(10x_1 + 17x_2 + 22x_3 3 + 21x_4) \\ 4x_1 + 5x_2 + 6x_3 + 2x_4 \leq 7 \\ x \in \{0,1\}^n \end{cases}$$

si hanno gli algoritmi di valutazione:

- **Valutazione inferiore:** si prendono le variabili con rendimenti migliori, una volta sola, finché non si satura. Nel caso la variabile esca da P, si prende quella dopo, con il caso limite di non prendere nulla. Si ha quindi:

$$x = (0, 1, 0, 1), \quad V_I = 21 + 17 = 38$$

- **Valutazione superiore:** si prende il rilassato continuo:

$$\begin{cases} \max(v^\intercal x) \\ p^\intercal x \leq \\ 0 \leq x \leq 1 \end{cases}$$

e si riempie con le variabili dai rendimenti migliori, saturando l'ultima:

$$x = \left(0, 0, \frac{5}{6}, 1\right), \quad V_\alpha = 21 + \frac{5}{6}22 = 39.\overline{3}, \quad V_S = \lfloor V_\alpha \rfloor = \lfloor 39.\overline{3} \rfloor = 39$$

Si ha quindi $V_I = 38$ e $V_S = 39$, con errore $\epsilon = \frac{39-38}{38} = \frac{1}{38} = 2.6\%$.

- **Problema intero**

$$\begin{cases} \max(10x_1 + 17x_2 + 22x_3 3 + 21x_4) \\ 4x_1 + 5x_2 + 6x_3 + 2x_4 \leq 7 \\ x \leq 0 \end{cases}$$

si hanno gli algoritmi di valutazione:

- **Valutazione inferiore:** si prendono le variabili con rendimenti migliori, con coefficienti maggiori possibili, finché non si satura. Nel caso la variabile esca da P, si prende quella dopo, con il caso limite di non prendere nulla. Si ha quindi:

$$x = (0, 0, 0, 3), \quad V_I = 3 \cdot 21 = 63$$

- **Valutazione superiore:** si prende il rilassato continuo:

$$\begin{cases} \max(v^\intercal x) \\ p^\intercal x \leq \\ 0 \leq 0 \end{cases}$$

e si riempie con le variabili dai rendimenti migliori, saturando dalla prima:

$$x = \left(0, 0, 0, \frac{7}{2}\right), \quad V_\alpha = \frac{7}{2}22 = 73.5, \quad V_S = \lfloor V_\alpha \rfloor = \lfloor 73.5 \rfloor = 73.5$$

Si ha quindi $V_I = 63$ e $V_S = 73.5$, con errore $\epsilon = \frac{73.5-63}{63} = \frac{1}{6} = 16\%$. Notiamo come sul problema intero si accumuli molto più errore.

Dobbiamo fare un'ulteriore precisazione: si prendono approssimazioni per **difetto** quanto si parla di problemi di *massimo*. Nel caso di problemi di *minimo*, è opportuno prendere approssimazioni per **eccesso**.

14 Lezione del 15-10-24

14.1 Relazioni tra LP e ILP

Vediamo di approfondire il legame fra un problema ILP e i problemi LP che possiamo ricavarne. Avevamo posto un problema ILP in forma:

$$\begin{cases} \max(c^\intercal x) \\ Ax \leq b \\ x \in \mathbb{Z}^n \end{cases}$$

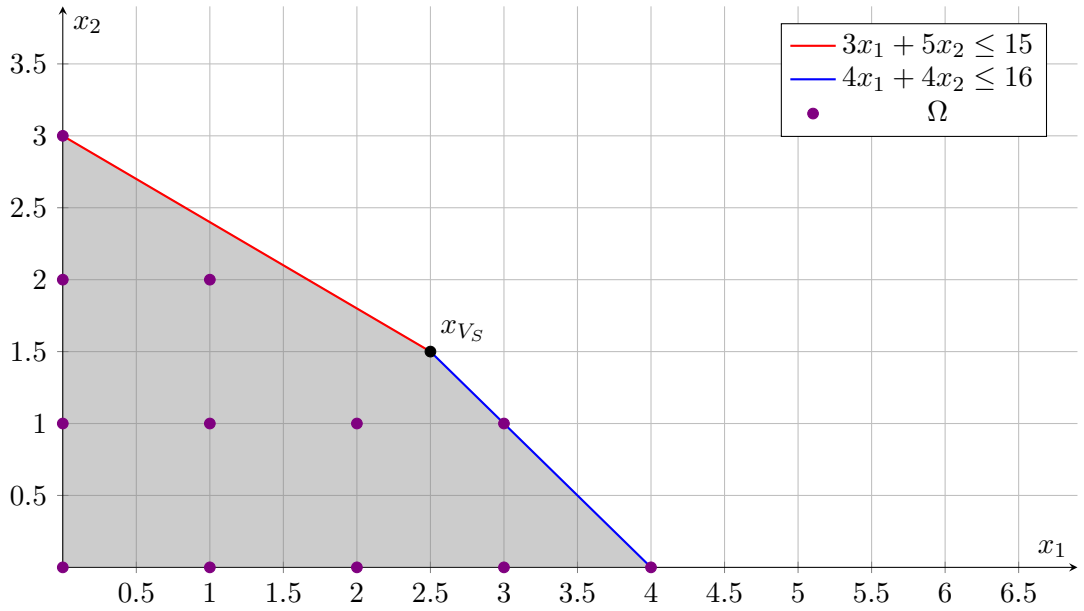
E avevamo visto che si può trovare un limite inferiore V_I e un limite superiore V_S a partire dagli algoritmi dei rendimenti, quindi prendendo il rilassato continuo del problema, cioè l'associato che rimuove il vincolo $x \in \mathbb{Z}^n$:

$$\begin{cases} \max(c^\intercal x) \\ Ax \leq b \end{cases}$$

Chiamiamo P il poliedro del rilassato continuo. Si ha, in generale, che la soluzione di un problema di ILP è uno dei punti $\in \Omega = P \cap \mathbb{Z}^n$, cioè dei punti $\in \mathbb{Z}^n$ che stanno all'interno del poliedro del rilassato continuo. Poniamo ad esempio il problema:

$$\begin{cases} \max(0.3x_1 + 0.4x_2) \\ 3x_1 + 5x_2 \leq 15 \\ 4x_1 + 4x_2 \leq 16 \\ x_i \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

Graficamente, si ha:



dove si è riportata la soluzione ottima del rilassato x_{V_S}.

Notiamo quindi che esiste un'insieme:

$$\text{conv}(\Omega) = \left\{ x \in \mathbb{R}^n : \exists x_1, ..., x_p \in \Omega, \quad \lambda_1, ..., \lambda_p \geq 0 \quad \text{t.c.} \quad x = \sum_{i=1}^{p} \lambda_i x_i, \quad \sum_{i=1}^{p} \lambda_i = 1 \right\}$$

cioè l'involucro convesso di Ω. Visto che i punti di Ω hanno componenti intere, si può dimostrare il seguente teorema:

Teorema 14.1: Caratterizzazione della regione ammissibile di un problema ILP

Dato un problema ILP con regione ammissibile Ω, esiste un insieme finito di punti $\{q_l\}_{l \in L} = \{q_1, ..., q_{|L|}\}$ di Ω, e un insieme finito di direzioni di recessione $\{r_j\}_{j \in J} = \{r_1, ..., r_|J|\}$ di P, tali che:

$$\Omega = \left\{ x \in R_+^n : x = \sum_{l \in L} \alpha_l q_l + \sum_{j \in J} \beta_j r_j, \quad \sum_{l \in L} \alpha_l = 1, \quad \alpha \in \mathbb{Z}_+^{|L|}, \quad \beta \in \mathbb{Z}_+^{|J|} \right\}$$

cioè si puo ricavare Ω attraverso una forma simile alla $P = \text{conv}(V) + \text{cono}(E)$ del teorema di Minkowski-Weyl, sugli insiemi $\{q_l\}_{l \in L}$ di vertici a componenti intere e $\{r_j\}_{j \in J}$ di direzioni di recessione.

A partire da questa caratterizzazione di Ω, vogliamo caratterizzare $\text{conv}(\Omega)$:

Teorema 14.2: Caratterizzazione dell'involucro convesso della regione ammissibile di un problema ILP

Dato un problema ILP con regione ammissibile Ω, si ha che $\text{conv}(\Omega)$ è un **poliedro razionale**, cioè esistono due insiemi finiti di vettori, $\{q_l\}_{l \in L}$ e $\{r_j\}_{j \in J}$, a **componenti razionali**, tali che:

$$\text{conv}(\Omega) = \text{conv}\,\{q_l\}_{l \in L} + \text{cono}\{r_j\}_{j \in J}$$

Addirittura, normalizzando si può supporre che gli r_j siano a componenti intere.

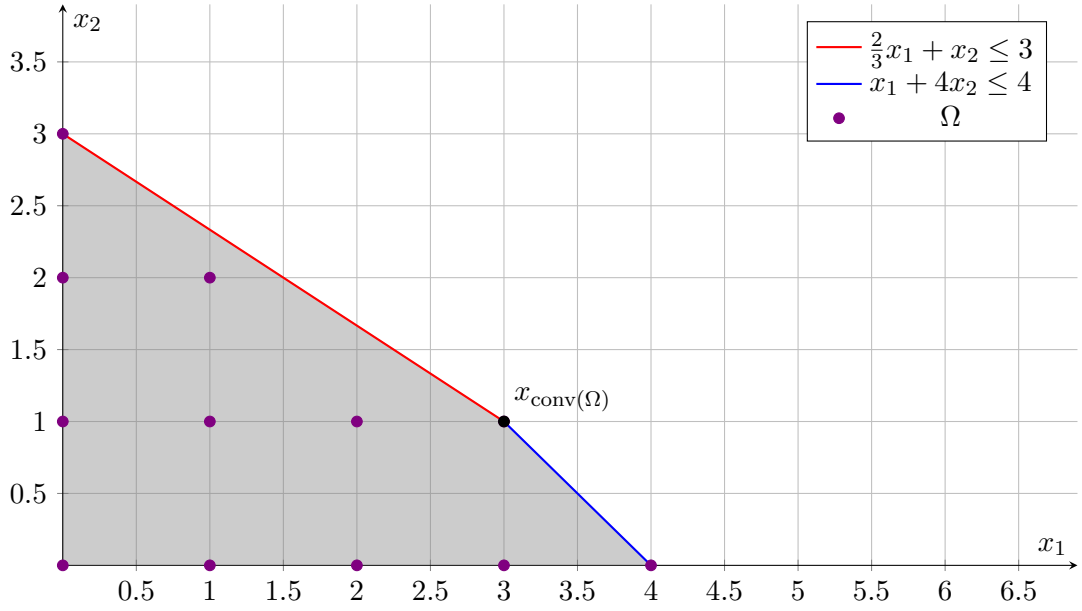
Nell'esempio precedente, $\text{conv}(\Omega)$ sarebbe rappresentato dagli insiemi:

$$\{q_l\}_{l \in L} = \{(0,0), (0,3), (3,1), (4,0)\}, \quad \{r_j\}_{j \in J} = \emptyset$$

quindi:

$$\text{conv}(\Omega) = \text{conv}\,\{q_l\}_{l \in L} + \text{cono}\{r_j\}_{j \in J} = (x_1, x_2) \in R^2 \quad \text{t.c.} \quad \begin{cases} \frac{2}{3}x_1 + x_2 \leq 3 \\ x_1 + x_2 \leq 4 \end{cases}$$

cioè sul grafico:



dove è stata evidenziata la soluzione del primale sull'insieme $\text{conv}(\Omega)$, $x_{\text{conv}(\Omega)}$.

Possiamo quindi dire, visto che $\Omega \subset \text{conv}(\Omega)$ e che P è un'estensione di Ω in quanto poliedro del rilassato continuo, che è vera la catena di diseguaglianze:

$$\max_{x \in \Omega} c^\intercal x \leq \max_{x \in \text{conv}(\Omega)} c^\intercal x \leq \max_{x \in P} c^\intercal x$$

e non solo: si può stringere la diseguaglianza sul lato sinistro, per affermare che:

Teorema 14.3: Equivalenza fra problemi LP e ILP

Si prenda un problema di ILP, e il problema di LP associato costruito su:

$$\text{conv}(\Omega) = \text{conv}\,\{q_l\}_{l \in L} + \text{cono}\{r_j\}_{j \in J}$$

con, posto P come il poliedro del rilassato continuo, q ricavato dai vertici $P \cap \mathbb{Z}^n$, e r ricavato dalle direzioni di recessione di P.
Se si prendono le soluzioni:

$$v_\Omega = \max_{x \in \Omega} c^\intercal x, \quad v_{\text{conv}(\Omega)} = \max_{x \in \text{conv}(\Omega)} c^\intercal x$$

si ha che $v_\Omega = v_{\text{conv}(\Omega)}$, e che se $v_{\text{conv}(\Omega)}$ è finito, allora esiste $x_\Omega \in \Omega$ tale che $c^\intercal x_\Omega = v_\Omega = v_{\text{conv}(\Omega)}$

Siamo quindi arrivati a formulare un teorema secondo cui, per ogni problema ILP, possamo costruire un problema LP associato che ha la stessa soluzione, semplicemente riformulando i vincoli in modo che descrivano l'involucro convesso dei punti in $\Omega = P \cap \mathbb{Z}^n$, ed eventuali direzioni di recessione di P, dove P è il poliedro del rilassato continuo. Il problema sorge dal fatto che è *difficile* (nell'accezione di *difficile* data dalla complessità) ricavare questo problema associato.

14.1.1 Caratterizzazione dell'involucro convesso

Esistono tre strade principali per il calcolo di $\text{conv}(\Omega)$:

- **Matrici unimodulari:** in problemi LP dove i vincoli sono espressi da matrici particolari, che chiameremo **unimodulari**, si ha che P ha vertici a componenti intere, e quindi il $\text{conv}(\Omega) = P \cap \mathbb{Z}^n$ degli Ω a componenti interi non potrà che coincidere con P che li contiene. Questi problemi sono ad esempio quelli di flusso a costo minimo.
- **Combinatorica poliedrale:** in alcuni problemi LP espressi su grafi, si può stabilire la forma delle disequazioni che esprimono $\text{conv}(\Omega)$. Ad esempio, si ha che i problemi di accoppiamento massimo sui grafi, che richiedono di trovare il maggior numero di archi attraverso il cui si può realizzare un accoppiamento, risulta coincidente in P e $\text{conv}\Omega$ quando il grafo di interesse è **bipartito**, cioè è divisibile in due insiemi di nodi U e V tali che ogni arco porta da $U \to V$ o viceversa.
- **Piani di taglio:** quando non si possono applicare i metodi riportati sopra, l'unica strada è quella di trovare una serie di disequazioni successive che riducono l'insieme P fino al minimale $\text{conv}(\Omega)$. Uno dei metodi che andremo a vedere per realizzare queste riduzioni è quello dei **piani di taglio di Gomory**.

14.1.2 Matrici unimodulari

Definiamo innanzitutto:

Definizione 14.1: Matrice unimodulare

Si chiama **modulare** ogni matrice quadrata intera con determinante $\det A_m \in \{1, -1\}$.

e, sulla base di questo:

Definizione 14.2: Matrice totalmente unimodulare

Si chiama **totalmente unimodulare** ogni matrice per cui ogni sottomatrice quadrata invertibile è unimodulare, cioè ogni sottomatrice quadrata ha determinante $\det(A_m) \in \{0, 1, -1\}$.

Si ha che se una matrice è unimodulare, allora i vertici della regione ammissibile appartengono a $\mathbb{Z}^n$, infatti:

Teorema 14.4: Soluzioni di base di matrici unimodulari

Dato $Ax \leq b$, se A e b sono a componenti intere, e A è totalmente unimodulare, allora tutte le soluzioni di base del poliedro P:

$$P = \{x \in \mathbb{R}^n : Ax \leq b\}$$

sono a componenti intere.

dove per componenti intere intendiamo anche razionali sotto normalizzazione.

Questo risulato è molto utile: abbiamo che le matrici A dei problemi di **trasporto** e **assegnamento di costo minimo** sono totalmente unimodulari, ergo possiamo risolvere le versioni ILP di quei problemi semplicemente rimuovendo il vincolo di interezza $x \in \mathbb{Z}^n$.

15 Lezione del 16-10-24

15.1 Problema di impacchettamento

Problema 15.1: Impacchettamento

Poniamo di avere 6 file, contenenti registrazioni di tutta la musica di Mozart, con il seguente ingombro in gigabyte:

	Arie	Opere	Concerti	Sinfonie	Sonate	Messe
p	3	6	5	4	4	8

Vogliamo trovare il numero minimo di dischi rigidi di dimensione $P = 10$ per archiviare tutti di questi file.

Chiamiamo questi problemi anche problemi di *bin-packing*. Rappresentiamo la soluzione come una matrice di adiacenza:

$$x_{ij} = \begin{cases} 0 \\ 1 \end{cases}$$

dove $i \in \{1, ..., p\}$ rappresenta il contenitore e $j \in \{1, ..., 6\}$ l'oggetto che vi inseriamo. Conviene trovare prima una stima superiore per il numero di contenitori p, attraverso un'opportuno algoritmo greedy.

Un'algoritmo banale può essere quello di riempire finché è possibile il primo contenitore, cioè finche si hanno oggetti che entrano nello spazio libero (detto *sfrido*) del contenitore. Una volta che questa ipotesi è violata, si prende un'altro contenitore, e così via.

Si ricava quindi questa valutazione superiore V_S. A questo punto si può porre:

$$\begin{cases} x_{11} + x_{21} + x_{31} + x_{41} = 1 \\ x_{12} + x_{22} + x_{32} + x_{42} = 1 \\ ... \\ x_{16} + x_{26} + x_{36} + x_{46} = 1 \end{cases}$$

cioè vogliamo prendere uno e uno solo di tutti gli oggetti, disposti fra i $V_S = 4$ contenitori di valutazione superiore.

Visto che non siamo sicuri di dover prendere tutti i contenitori, dovremo introdurre una variabile y_i per ognuno di essi:

$$y_i = \begin{cases} 0 \\ 1 \end{cases}$$

Infine vogliamo inserire la dimensione di ogni contenitore, $P = 10$, nel problema, sulla base dei pesi p_i di ogni oggetto:

$$\begin{cases} p_1 x_{11} + p_2 x_{12} + p_3 x_{13} + p_4 x_{14} + p_5 x_{15} + p_6 x_{16} \leq 10 y_1 \\ p_1 x_{21} + p_2 x_{22} + p_3 x_{23} + p_4 x_{24} + p_5 x_{25} + p_6 x_{26} \leq 10 y_2 \\ ... \\ p_1 x_{41} + p_2 x_{42} + p_3 x_{43} + p_4 x_{44} + p_5 x_{45} + p_6 x_{46} \leq 10 y_4 \end{cases}$$

Combinando quanto posto finora, otteniamo il problema completo:

$$\begin{cases} \min(y_1 + y_2 + y_3 + y_4) \\ x_{11} + x_{21} + x_{31} + x_{41} = 1 \\ x_{12} + x_{22} + x_{32} + x_{42} = 1 \\ ... \\ x_{16} + x_{26} + x_{36} + x_{46} = 1 \\ p_1x_{11} + p_2x_{12} + p_3x_{13} + p_4x_{14} + p_5x_{15} + p_6x_{16} \leq 10y_1 \\ p_1x_{21} + p_2x_{22} + p_3x_{23} + p_4x_{24} + p_5x_{25} + p_6x_{26} \leq 10y_2 \\ ... \\ p_1x_{41} + p_2x_{42} + p_3x_{43} + p_4x_{44} + p_5x_{45} + p_6x_{46} \leq 10y_4 \\ x_i \in \{0, 1\} \\ y_i \in \{0, 1\} \end{cases}$$

dove minimizziamo gli y_1 cercando di usare meno contenitori possibile. Questo è un problema di ILP. Cerchiamo quindi la valutazione inferiore V_I e la superiore V_S.

Possiamo dare una stima inferiore attraverso la formula:

$$V_I = \left\lceil \frac{\sum_{j=1}^{6} p_i}{P} \right\rceil$$

cioè il peso di tutti gli oggetti fratto le dimensioni dei contenitori, arrotondato per eccesso, che è il minimo numero di contenitori possibile per contenere tutti gli oggetti.

Per il calcolo della stima superiore, invece, avevamo presentato un algoritmo greedy. In verità sono ci altre (e più intelligenti) strade che possiamo prendere:

- **Next-fit decreasing:** essenzialmente l'algoritmo presentato, dove si ha:

Algoritmo 5 next-fit decreasing per impachettamento

```
Input: un problema di impacchettamento
Output: una soluzione ammissibile
while ci sono ancora oggetti do
    Prendi il prossimo oggetto
    if entra nel contenitore then
        Inseriscilo nel contenitore
    else
        Prendi un'altro contenitore e inseriscici l'oggetto
    end if
end while
```

- **First-fit decreasing:** analogo al next-fit, ma con la differenza che per ogni oggetto si considerano tutti i contenitori:

Algoritmo 6 first-fit decreasing per impachettamento

Input: un problema di impacchettamento
Output: una soluzione ammissibile
while ci sono ancora oggetti **do**
  Prendi il prossimo oggetto
  if l'oggetto entra in uno dei contenitori presi finora **then**
    Inseriscilo nel contenitore
  else
    Prendi un'altro contenitore e inseriscici l'oggetto
  end if
end while

- **Best-fit decreasing:** è una variante del first-fit che prende sempre i contenitori con sfrido massimo, cioè cerca di riempire i contenitori con meno spazio disponibile (cioè di trovare l'"incastro" migliore per l'oggetto):

Algoritmo 7 best-fit decreasing per impachettamento

Input: un problema di impacchettamento
Output: una soluzione ammissibile
while ci sono ancora oggetti **do**
  Prendi il prossimo oggetto
  if l'oggetto entra in uno dei contenitori, ordinati per sfrido decrescente, presi finora **then**
    Inseriscilo nel contenitore
  else
    Prendi un'altro contenitore e inseriscici l'oggetto
  end if
end while

16 Lezione del 17-10-24

16.1 Piani di taglio

Abbiamo visto finora il problema:

$$\begin{cases} \max c^\intercal x \\ Ax \leq bx \in \mathbb{Z}^n_+ \end{cases}$$

Dove si può definire il poliedro P del rilassato continuo:

$$P = \{x \in \mathbb{R}^n : Ax \leq b\} \to s_{PL},\ x_{RC}$$

L'insieme dei punti Ω:

$$\Omega = \{x \in \mathbb{Z}^n : Ax \leq b\} \to v_{PLI},\ \bar{x}$$

E un problema di PL associato, "ristretto" sul convesso dell'insieme Ω, di cui però non sapevamo caratterizzare le disequazioni (salvo alcuni casi specifici in problemi di PL sui grafi):

$$\text{conv}(\Omega) \to v_{\text{conv}(\Omega)},\ \bar{x}$$

Sappiamo valere, fra queste soluzioni, la catena di diseguaglianze:

$$v_{PLI} = v_{\text{conv}(\Omega)} \leq v_{PL}$$

Decidiamo di ridurre le dimensioni dell'insieme P, visto che $\text{conv}(\Omega) \subseteq P$, per approssimare $\text{conv}(\Omega)$ stesso. Facciamo ciò sfruttando le cosiddette **disuguaglianze valide** (DV):

Definizione 16.1: Diseguaglianza valida

Una diseguaglianza è una DV di un'insieme $\Omega \subset \mathbb{Z}^n$ se rispetta la forma:

$$\gamma^\intercal x \leq \gamma_0, \quad \forall x \in \Omega$$

Abbiamo quindi che una diseguaglianza è valida se contiene Ω completamente nel semispazio che definisce. Si può poi dimostrare il fatto piuttosto ovvio:

Teorema 16.1: Diseguaglianza valida dell'involucro convesso

una diseguaglianza valida per un'insieme Ω lo è anche per $\text{conv}(\Omega)$.

Una tecnica di base per il calcolo delle diseguaglianze valide è quella dell'approssimazione per difetto delle diseguaglianze che definiscono P. Questo però è poco utile nel caso di problemi già definiti con diseguaglianze in componenti intere. Introduciamo quindi l'idea di un **piano di taglio**:

Definizione 16.2: Piano di taglio

Su un problema di PLI con soluzione x_{RC} al rilassato continuo, una DV in forma:

$$\gamma x \leq \gamma_0, \quad \forall x \in S \ : \ \gamma x_{RC} > \gamma_0$$

si dice **piano di taglio**.

L'idea dei piani di taglio è quella di rimuovere successivamente soluzioni di P fuori da $\text{conv}(\Omega)$. Supponiamo di avere il problema:

$$\begin{cases} \max c^\intercal x \\ Ax \leq b \\ \gamma x \leq \gamma_0 \end{cases} \rightarrow V_{P_n}$$

Vogliamo ricavare un poliedro P_n, che contiene l'ottimo v_{PLI}, ma che è più piccolo del rilassato continuo:

$$v_{PLI} \leq v_{P_n} \leq v_{PL}$$

Per fare ciò, prendiamo una DV che non contiene x_{RC} soluzione del rilassato continuo. A questo punto si avrà che, se il problema di v_{P_n} ha soluzione a componenti intere, allora coinciderà con $v_{\text{conv}(\Omega)}$, altrimenti si dovranno ripetere i passi precedenti.

Abbiamo quindi che qualsiasi problema trovato iterativamente coi piani di taglio è una riduzione della regione ammissibile. Il sottoinsieme dei piani di taglio che studieremo è quello dei **piani di taglio di Gomory**.

16.1.1 Costruzione dei piani di taglio di Gomory

Prendiamo un problema di ILP in forma:

$$\begin{cases} \max c^\intercal x \\ Ax = b \\ x \geq 0 \\ x \in \mathbb{Z}^n \end{cases}$$

cioè in una forma simile al duale standard. Possiamo convertire qualsiasi problema in questa forma introducendo eventuali variabili di surplus per portare le disequazioni in equazioni. Ad esempio, sul problema presentato un paio di lezioni fa:

$$\begin{cases} \max(0.3x_1 + 0.4x_2) \\ 3x_1 + 5x_2 \leq 15 \\ 4x_1 + 4x_2 \leq 16 \\ x_i \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

si ricava:

$$\begin{cases} \max(0.3x_1 + 0.4x_2) \\ 3x_1 + 5x_2 + x_3 = 15 \\ 4x_1 + 4x_2 + x_4 = 16 \\ x_i \geq 0 \\ x \in \mathbb{Z}^2 \end{cases}$$

Dalla soluzione del rilassato continuo x_{RC}, abbiamo una base ottima B, tale per cui si può porre:

$$A = (A_B \; A_N), \quad x_{RC} = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$$

cioè, A_B sono le colonne di matrice corrispondenti alla base e A_N le altre, e x_B sono le variabili di base e x_N le altre.

Definiamo allora la matrice $\tilde{A}$:

$$\tilde{A} = A_B^{-1} A_N$$

Prendiamo quindi un indice $r \in B$, tale che x_{RC} a r ha una componente non intera. Possiamo a questo punto enunciare il teorema:

Teorema 16.2: Teorema di Gomory

Si ha che:

$$\sum_{j \in N} \{\tilde{a}_{rj}\} x_j \geq \{(x_B)_r\}$$

è un piano di taglio.

Occorre fare qualche chiarimento sulla notazione: le graffe rappresentano l'operatore componente frazionaria, mentre l'r a pedice di x_B indica di prendere la componente all'indice r, cioè quella non intera.

Applichiamo quindi questo metodo al problema di prima. Avremo che:

$$A = \begin{pmatrix} 3 & 5 & 1 & 0 \\ 4 & 4 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 15 \\ 16 \end{pmatrix}$$

e la soluzione ottima $x_{RC} = \left(\frac{5}{2}, \frac{3}{2}\right)$ sulla base ottima $B = \{1, 2\}$. Abbiamo quindi che:

$$A_B = \begin{pmatrix} 3 & 5 \\ 4 & 4 \end{pmatrix}, \quad A_N = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad x_B = \begin{pmatrix} \frac{5}{2}, \frac{3}{2} \end{pmatrix}, \quad x_N = (0, 0)$$

e la matrice $\tilde{A}$ è:

$$\tilde{A} = A_B^{-1} A_N = A_B^{-1} = \begin{pmatrix} -\frac{1}{2} & \frac{5}{8} \\ \frac{1}{2} & -\frac{3}{8} \end{pmatrix}$$

Notiamo che in x_B entrambi i componenti sono non interi, ergo possiamo prendere i due casi:

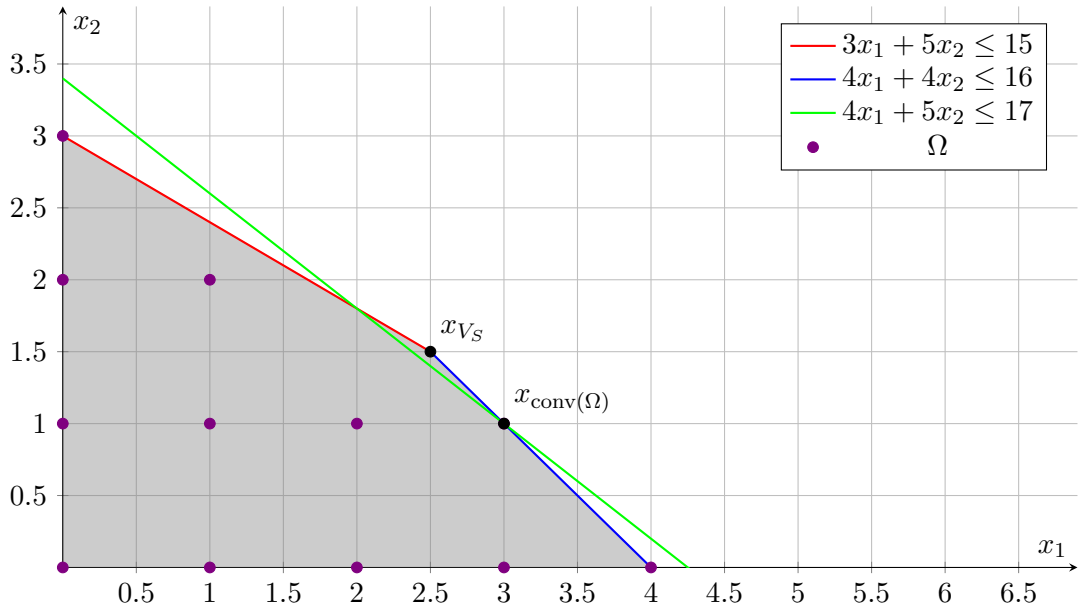
- $r = 1$, si ha $\left\{-\frac{1}{2}\right\} x_3 + \left\{\frac{5}{8}\right\} x_4 \geq \left\{\frac{5}{2}\right\}$. Applicando l'operatore parte frazionaria, e sostituendo a x_3 e x_4 le loro espressioni in funzione di x_1 e x_2 (ricavate dalle equazioni del poliedro) si ha:

$$4x_1 + 5x_2 \leq 17$$

- $r = 2$, si ha con calcoli simili, lo stesso taglio:

$$4x_1 + 5x_2 \leq 17$$

Quindi entrambe le componenti danno lo stesso taglio (si noti che questa non è la norma). Possiamo visualizzare sul grafico cosa significa il taglio introdotto:



Notiamo che il taglio, se non ha ridotto completamente il poliedro a $\text{conv}(\Omega)$, ha portato la soluzione ottima a quella di $\text{conv}(\Omega)$.

17 Lezione del 21-10-24

17.1 Applicazione dell'algoritmo di riduzione del gap

Applichiamo il metodo dei piani di taglio di Gomory ad alcuni problemi di ILP.

17.1.1 Zaino intero

Prendiamo il problema:

$$\begin{cases} \max 30x_1 + 36x_2 + 27x_3 + 20x_4 + 24x_5 + 22x_6 \\ 13x_1 + 16x_2 + 14x_3 + 15x_4 + 17x_5 + 14x_6 \leq 57 \\ x \in \mathbb{Z}_+ \end{cases}$$

con il vettore dei rendimenti:

$$r = (2.3, 2.25, 1.93, 1.33, 1.41, 1.57)$$

Saturiamo per ottenere una soluzione x_{RC}, e quindi una valutazione superiore:

$$x_{RC} = \left(\frac{57}{13}, 0, 0, 0, 0, 0\right), \quad v_S = 131$$

e saturiamo l'intero per avere una valutazione inferiore:

$$x_I = (4, 0, 0, 0, 0, 0), \quad v_I = 120$$

Adesso applichiamo Gomory:

1. Si porta il rilassato continuo in formato duale standard:

$$\begin{cases} \max 30x_1 + 36x_2 + 27x_3 + 20x_4 + 24x_5 + 22x_6 \\ 13x_1 + 16x_2 + 14x_3 + 15x_4 + 17x_5 + 14x_6 + x_7 = 57 \\ x_i \geq 0 \end{cases}$$

2. Si individua la base ottima:

$$x_{RC} = \left(\frac{57}{13}, 0, 0, 0, 0, 0\right) \Rightarrow B = \{1\}$$

3. Si ricavano A_B, A_N, x_B, x_N e r:

$$A_B = \begin{pmatrix} 13 \end{pmatrix}, \quad A_N = \begin{pmatrix} 16 & 14 & 15 & 17 & 14 & 1 \end{pmatrix}, \quad x_B = \begin{pmatrix} \frac{57}{13} \end{pmatrix}, \quad x_N = (0, 0, 0, 0, 0)$$

e chiaramente $r = 1$;

4. Si ricava $\tilde{A}$:

$$\tilde{A} = A_B^{-1} A_N = \left(\frac{16}{13}, \frac{14}{13}, \frac{15}{13}, \frac{17}{13}, \frac{14}{13}, \frac{1}{13}\right)$$

5. Si trova il nuovo vincolo:
$$\left\{\frac{16}{13}\right\}x_2+\left\{\frac{14}{13}\right\}x_3+\left\{\frac{15}{13}\right\}x_4+\left\{\frac{17}{13}\right\}x_5+\left\{\frac{14}{13}\right\}x_6+\left\{\frac{1}{13}\right\}x_7\geq\left\{\frac{57}{13}\right\}$$
$$\frac{3}{13}x_2+\frac{1}{13}x_3+\frac{2}{13}x_4+\frac{4}{13}x_5+\frac{1}{13}x_6+\frac{1}{13}x_7\geq\frac{5}{13}$$
Che espresso sostituendo $x_7=57-13x_1-16x_2-14x_3-15x_4-17x_5-14x_6$ e unito a quanto già trovato dà il problema:
$$\begin{cases}\max 30x_1+36x_2+27x_3+20x_4+24x_5+22x_6\\13x_1+16x_2+14x_3+15x_4+17x_5+14x_6\leq 57\\x_1+x_2+x_3+x_4+x_5+x_6\leq 4\\x\in\mathbb{Z}_+\end{cases}$$
da cui $x_{RC}=\left(\frac{7}{3},\frac{5}{3},0,0,0,0\right)$, che è già più vicino all'ottimo $\bar{x}=(3,1,0,0,0,0)$.

17.1.2 Zaino booleano

Prendiamo a questo punto:
$$\begin{cases}\max 30x_1+36x_2+27x_3+20x_4+24x_5+22x_6\\13x_1+16x_2+14x_3+15x_4+17x_5+15x_6\leq 57\\x\in\mathbb{Z}_+\\0\leq x\leq 1\end{cases}$$

Attraverso la saturazione, troviamo:
$$x_{RC}=\left(1,1,1,0,0,\frac{14}{15},0\right),\quad v_S=115.4$$
di valutazione superiore, e
$$x_I=(1,1,1,0,0,0),\quad v_I=93$$
di valutazione inferiore.

Riapplichiamo Gomory.

1. L'ottimo è noto:
$$x_{RC}=\left(1,1,1,0,0,\frac{14}{15},0\right)$$
2. Convertiamo in formato duale standard, ricordando i vincoli $0\leq x_i\leq 1$:
$$\begin{cases}\max 30x_1+36x_2+27x_3+20x_4+24x_5+22x_6\\13x_1+16x_2+14x_3+15x_4+17x_5+15x_6+x_7=57\\x_1+x_8=1\\x_2+x_9=1\\x_3+x_{10}=1\\x_4+x_{11}=1\\x_5+x_{12}=1\\x_6+x_{13}=1\\x\geq 0\end{cases}$$

Dobbiamo ricalcolare l'ottimo:

$$x'_{RC} = \left(1, 1, 1, 0, 0, \frac{14}{15}, 0, 0, 0, 0, 1, 1, \frac{1}{15}\right)$$

3. Si individua la base ottima:

$$x'_{RC} = \left(1, 1, 1, 0, 0, \frac{14}{15}, 0, 0, 0, 0, 1, 1, \frac{1}{15}\right) \Rightarrow B = \{1, 2, 3, 6, 11, 12, 13\}$$

4. Si ricavano A_B, A_N, x_B, x_N e r: Conviene prima scrivere A:

$$A = \begin{pmatrix} 13 & 16 & 14 & 15 & 17 & 15 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

e quindi:

$$A_B = \begin{pmatrix} 13 & 16 & 14 & 15 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}, \quad A_N = \begin{pmatrix} 15 & 17 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$x_B = \left(1, 1, 1, \frac{14}{15}, 1, 1, \frac{1}{15}\right), \quad x_N = (0, 0, 0, 0, 0, 0)$$

e $r = 4, 7$;

5. Si ricava $\tilde{A}$:

$$\tilde{A} = A_B^{-1} A_N = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & \frac{17}{15} & \frac{1}{15} & -\frac{13}{15} & -\frac{16}{15} & -\frac{14}{15} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & -\frac{17}{15} & -\frac{1}{15} & \frac{13}{15} & \frac{16}{15} & \frac{14}{15} \end{pmatrix}$$

6. Si trovano due nuovi vincoli, notando che $N = \{4, 5, 7, 8, 9, 10\}$:

$$r = 4 \Rightarrow x_1 + 2x_2 + x_3 + x_4 + x_5 + x_6 \leq 4$$

$$r = 7 \Rightarrow 13x_1 + 15x_2 + 14x_3 + 14x_4 + 15x_5 + 14x_6 \leq 55$$

17.1.3 Produzione

Supponiamo di evere il problema di minimazzazione:

$$\begin{cases} \max 5x_1 + 14x_2 \\ 16x_1 + 13x_2 \geq 62 \\ 6x_1 + 15x_2 \geq 52 \\ x \in \mathbb{Z}^2_+ \end{cases}$$

Possiamo fare le valutazioni:

$$x_{RC} = \left(\frac{26}{3}, 0\right), \quad v_I = 43$$

$$x_S = (9, 0), \quad v_S = 45$$

dove notiamo che l'arrotondamento della soluzione del rilassato è stavolta **per eccesso**, e rappresenta la **valutazione superiore**, in quanto il problema è **minimizzante**.

Applichiamo quindi Gomory:

1. Convertiamo in formato duale standard:

$$\begin{cases} \max 5x_1 + 14x_2 \\ 16x_1 + 13x_2 - x_3 = 62 \\ 6x_1 + 15x_2 - x_4 = 52 \\ x \geq 0 \end{cases}$$

   Dobbiamo ricalcolare l'ottimo per rendere conto alla variabile x_3 introdotta:

$$x'_R C = \left(\frac{26}{3}, 0, \frac{220}{3}, 0\right)$$

2. Si individua la base ottima dall'ottimo ricalcolato:

$$x'_R C = \left(\frac{26}{3}, 0, \frac{220}{3}, 0\right) \Rightarrow B = \{1, 3\}$$

3. Si ricavano A_B, A_N, x_B, x_N e r:

$$A_B = \begin{pmatrix} 16 & -1 \\ 6 & 0 \end{pmatrix}, \quad A_N = \begin{pmatrix} 13 & 0 \\ 15 & -1 \end{pmatrix}, \quad x_B = \left(\frac{26}{3}, \frac{230}{3}\right), \quad x_N = (0, 0)$$

   e $r = 1, 3$.

4. Si ricava $\tilde{A}$:

$$\tilde{A} = A_B^{-1} A_N = \begin{pmatrix} \frac{5}{2} & -\frac{1}{6} \\ 27 & -\frac{8}{3} \end{pmatrix}$$

5. Si ricavano i due nuovi vincoli:

$$r = 1 \Rightarrow 5x_1 + 14x_2 \geq 44$$

$$r = 2 \Rightarrow 2x_1 + 5x_2 \geq 18$$

   dove si ricorda si sono sostituite le variabili di indici $N = \{2, 4\}$ con quanto ricavato dalla forma duale standard.

18 Lezione del 22-10-24

18.1 Problema del commesso viaggiatore

Problema 18.1: del commesso viaggiatore

Supponiamo che un commesso viaggiatore debba fare il giro di 5 città, passando da tutte una e una sola volta. Si prepara una tabella con i tempi di percorrenza fra le città:

	Roccalbegna	Cana	Vallerona	Santa Caterina	Triana
Roccalbegna	-	18	14	17	19
Cana	16	-	19	22	23
Vallerona	17	14	-	18	20
Santa Caterina	16	19	22	-	21
Triana	15	14	13	20	-

Quale percorso dovrà seguire il commerciante, in modo da minimizzare la distanza percorsa?

Il problema del commesso viaggiatore (in inglese *Traveling Salesman Problem*, TSP) è effettivamente quello di trovare un **ciclo hamiltioniano**, cioè ciclo su un grafo che passa da ogni nodo una e una sola volta. Possiamo innanzitutto porre la matrice di adiacenza:

$$C = \begin{pmatrix} - & 18 & 14 & 17 & 19 \\ 16 & - & 19 & 22 & 23 \\ 17 & 14 & - & 18 & 20 \\ 16 & 19 & 22 & - & 21 \\ 15 & 14 & 13 & 20 & - \end{pmatrix}$$

Notiamo che si può distinguere fra TSP **simmetrici** e **asimmetrici**, in base alla simmetria della matrice di adiacenza. Si ha che i due tipi di problema hanno algoritmi risolutivi molto diversi. In questo caso, come è chiaro dalla matrice, considereremo il caso **asimmetrico**.

Inoltre, senza togliere dalla generalità della trattazione, possiamo assumere tutte le connessioni come stabilite, quindi la matrice **completa**. In questo caso gli archi mancanti del grafo saranno rappresentati da un costo infinito nella matrice di adiacenza (compresi gli archi $x_{ij} \to x_{ij}$, cioè da un nodo allo stesso nodo).

Abbiamo che tutti i percorsi possibili, per n città partendo dalla prima, sono $(n-1)!$, e che il problema si dimostra NP-completo. Diventa inapplicabile un'algoritmo di enumerazione completa: dobbiamo trovare quindi un modello matematico su cui applicare i macchinari della PL. Chiamiamo C la matrice dei costi, a_{ij} l'arco generico, C_H il ciclo hamiltioniano e introduciamo la variabile binaria:

$$x_{ij} = \begin{cases} 0, & a_{ij} \notin C_H \\ 1, & a_{ij} \in C_H \end{cases}$$

che rappresenterà matrice di adiacenza rappresentante il percorso scelto. Come sempre, linearizzamo le matrici in vettori ordinati lessicograficamente, quindi:

$$C = \begin{pmatrix} x_{11} & x_{12} & ... & x_{1N} \\ x_{21} & x_{22} & ... & x_{2N} \\ ... & & & \\ x_{N1} & x_{N2} & ... & x_{NN} \end{pmatrix} \rightarrow c = (x_{11}, x_{12}, ..., x_{1N}, x_{21}, x_{22}, ..., x_{2N}, ..., x_{N1}, x_{N2}, ..., x_{NN})$$

A questo punto possiamo impostare il problema di ILP:

$$\begin{cases} \min c^\intercal x \\ x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = 1 \\ x_{21} + x_{22} + x_{23} + x_{24} + x_{25} = 1 \\ ... \\ x_{51} + x_{52} + x_{53} + x_{54} + x_{55} = 1 \\ x_{11} + x_{21} + x_{31} + x_{41} + x_{51} = 1 \\ x_{12} + x_{22} + x_{32} + x_{42} + x_{52} = 1 \\ .... \\ x_{15} + x_{25} + x_{35} + x_{45} + x_{55} = 1 \\ x \in \{0, 1\} \end{cases}$$

I primi $n = 5$ vincoli impongono che da ogni nodo esca uno e uno solo arco, mentre i seguenti $n = 5$ impongono che su ogni nodo arrivi uno e un solo arco. L'ultimo vincolo rappresenta il dominio booleano delle variabili (un arco o è incluso o non è incluso nel ciclo hamiltoniano).

Notiamo che questo non basta a rappresentare cicli hamiltoniani: sono infatti ammessi sottocicli disgiunti fra di loro. Aggiungiamo quindi i vincoli, detti **di connessione**, dato $S \subset N$ sottoinsieme qualsiasi dei nodi:

$$\sum_{i \in S \, j \notin S} x_{ij} \geq 1, \quad \forall S \subset N, \quad S \neq \emptyset$$

Questi vincoli rappresentano l'obbligo per ogni sottoinsieme S non vuoto di nodi di avere almeno un'arco uscente, così da evitare cicli isolati. Si prende sottoinsieme stretto in quanto sarebbe inutile chiedere un'arco uscente dall'insieme completo dei nodi. Notiamo inoltre che con $|S| = 1$, questo dà i primi vincoli del problema già posto (in forma $x_{11} + x_{12} + x_{13}$...), che risulta in quanto sarebbero gli archi uscenti da gruppi di cardinalità 1, cioè singoli nodi, e il singolo arco uscente da ogni singolo nodo è già una prerogativa dell'assegnamento.

Possiamo quindi dire, in modo più completo, che si prendono i vincoli:

$$\begin{cases} \sum_{j \in N, j \neq i} x_{ij} = 1, \quad \forall i \in N \\ \sum_{i \in N, i \neq j} x_{ij} = 1, \quad \forall j \in N \\ \sum_{i \in S, j \notin S} x_{ij} \geq 1, \quad \forall S \subset N, \quad 2 \leq |S| \leq n - 1 \end{cases}$$

18.1.1 Cardinalità dei vincoli

Si ha che i vincoli iniziali erano $2n$, n per gli archi entranti e n per gli archi uscenti. Inoltre, posto $2 \leq |S| \leq n - 1$, abbiamo che il numero di vincoli di connessione è:

$$|\mathcal{V}_{connessione}| = 2^n - n - 2$$

Ergo si ha un numero di vincoli pari a:

$$|\mathcal{V}| = 2^n - n - 2 + 2n = 2^n + n - 2$$

Si ha che i vincoli crescono quindi in maniera esponenziale. Ad esempio, ecco l'insieme completo dei vincoli preso $n = 5$, come dall'esempio precedente, generati attraverso un programma al computer:

$$\begin{cases}
x_{12} + x_{13} + x_{14} + x_{15} = 1 \\
x_{21} + x_{23} + x_{24} + x_{25} = 1 \\
x_{31} + x_{32} + x_{34} + x_{35} = 1 \\
x_{41} + x_{42} + x_{43} + x_{45} = 1 \\
x_{51} + x_{52} + x_{53} + x_{54} = 1 \\
x_{21} + x_{31} + x_{41} + x_{51} = 1 \\
x_{12} + x_{32} + x_{42} + x_{52} = 1 \\
x_{13} + x_{23} + x_{43} + x_{53} = 1 \\
x_{14} + x_{24} + x_{34} + x_{54} = 1 \\
x_{15} + x_{25} + x_{35} + x_{45} = 1 \\
x_{12} + x_{32} + x_{42} + x_{52} \geq 1 \\
x_{13} + x_{23} + x_{43} + x_{53} \geq 1 \\
x_{14} + x_{24} + x_{34} + x_{54} \geq 1 \\
x_{15} + x_{25} + x_{35} + x_{45} \geq 1 \\
x_{21} + x_{31} + x_{41} + x_{51} \geq 1 \\
x_{14} + x_{15} + x_{24} + x_{25} + x_{34} + x_{35} \geq 1 \\
x_{13} + x_{15} + x_{23} + x_{25} + x_{43} + x_{45} \geq 1 \\
x_{13} + x_{14} + x_{23} + x_{24} + x_{53} + x_{54} \geq 1 \\
x_{13} + x_{14} + x_{15} + x_{23} + x_{24} + x_{25} \geq 1 \\
x_{12} + x_{15} + x_{32} + x_{35} + x_{42} + x_{45} \geq 1 \\
x_{12} + x_{14} + x_{32} + x_{34} + x_{52} + x_{54} \geq 1 \\
x_{12} + x_{14} + x_{15} + x_{32} + x_{34} + x_{35} \geq 1 \\
x_{12} + x_{13} + x_{42} + x_{43} + x_{52} + x_{53} \geq 1 \\
x_{12} + x_{13} + x_{15} + x_{42} + x_{43} + x_{45} \geq 1 \\
x_{12} + x_{13} + x_{14} + x_{52} + x_{53} + x_{54} \geq 1 \\
x_{21} + x_{25} + x_{31} + x_{35} + x_{41} + x_{45} \geq 1 \\
x_{21} + x_{24} + x_{31} + x_{34} + x_{51} + x_{54} \geq 1 \\
x_{21} + x_{24} + x_{25} + x_{31} + x_{34} + x_{35} \geq 1 \\
x_{21} + x_{23} + x_{41} + x_{43} + x_{51} + x_{53} \geq 1 \\
x_{21} + x_{23} + x_{25} + x_{41} + x_{43} + x_{45} \geq 1 \\
x_{21} + x_{23} + x_{24} + x_{51} + x_{53} + x_{54} \geq 1 \\
x_{31} + x_{32} + x_{41} + x_{42} + x_{51} + x_{52} \geq 1 \\
x_{31} + x_{32} + x_{35} + x_{41} + x_{42} + x_{45} \geq 1 \\
x_{31} + x_{32} + x_{34} + x_{51} + x_{52} + x_{54} \geq 1 \\
x_{41} + x_{42} + x_{43} + x_{51} + x_{52} + x_{53} \geq 1
\end{cases}$$

Dobbiamo poi considerare il vincolo di interezza $x \in \mathbb{Z}$, e nel caso si prenda il rilassato continuo (che vedremo in questo caso è il problema di assegnamento corrispondente), i $2n$ vincoli in forma $x_{ij} \leq 1$, $x_{ij} \geq 0$.

18.1.2 Valutazioni inferiori e superiori

Vediamo come trovare buone valutazioni inferiori e superiori delle soluzioni v_{TSP} di problemi TSP asimmetrici.

- **Valutazione inferiore:** per fare una valutazione inferiore v_I, si rimuovono i vincoli di connessione, trasformando il problema in un problema di assegnamento (infatti si indica anche $v_I = v_{ASS}$) e concedendo quindi cicli disgiunti. Questo approccio è solitamente molto efficace per i problemi TSP *asimmetrici*, restituendo rapporti $\epsilon = \frac{v_{TSP} - v_{ASS}}{v_{ASS}}$ nell'ordine di poche frazioni di punto percentuale. Nel caso di TSP simmetrici, invece, il rilassamento restituisce errori fino al 60%, e non è quindi particolarmente indicato.
- **Valutazione superiore:** per fare una valutazione superiore v_S, invece, si usa il cosiddetto **algoritmo delle toppe**, o *algoritmo di fusione dei cicli disgiunti* (ma tu vedi gli americani). Questo algoritmo prevede di calcolare il v_I, quindi la soluzione che concede cicli disgiunti, e di selezionare da due di questi un arco ciascuno, (i, j) e (k, l). Si eliminano quindi questi archi, incrociandoli, cioè rimuovendo (i, j) e (k, l) e introducendo (i, l) e (k, j). Avremo variazione di v_{ASS}:

  $$v'_S = v_{ASS} - c_{ij} - c_{kl} + c_{il} + c_{kj}, \quad \Delta v = -c_{ij} - c_{kl} + c_{il} + c_{kj}$$

  Si ripete iterativamente il processo, riducendo di volta in volta il numero di vincoli, finche non si arriva ad un ciclo hamiltoniano. L'algoritmo completo è quindi:

Algoritmo 8 delle toppe

Input: una v_{ASS} soluzione dall'assegnamento ricavato da un TSP
Output: una valutazione inferiore v_I del TSP
ciclo:
Chiama $C = \{C_1, ..., C_p\}$ l'insieme dei cicli (sottocicli) dati da v_{ASS}
Per ogni coppia di cicli (C_h, C_k) valuta l'incremento di costo Δv dovuto alla fusione di C_h e C_k nel modo più conveniente possibile
Effettua la fusione dei cicli C_h e C_k con $\Delta v = \min_{C_i, C_j} \Delta v, \quad i, j \in [1, p], \quad i \neq j$
$p \leftarrow p - 1$
if $p > 1$ **then**
  Vai a ciclo
end if

19 Lezione del 23-10-24

19.1 Multizaino

Abbiamo visto in precedente i cosiddetti problemi di **caricamento**, o di *zaino*, cioè problemi in forma:

$$\begin{cases} \max \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x_i \in \{0, 1\} \end{cases}$$

dove gli c_i definiscono i **valori** e a_i i **pesi** di n oggetti, in questo caso presi una volta sola. Avevamo ammesso anche il caso con un numero arbitrario di oggetti:

$$\begin{cases} \max \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ x_i \geq 0 \end{cases}$$

Entrambi questi tipi di problemi si risolvono con valutazioni e algoritmi euristici, tenendo conto del vettore dei **rendimenti** $r_i = \frac{c_i}{a_i}$.

Esiste una variante del problema dove abbiamo più di un vincolo di peso: questa si dice **multizaino**. In forma binaria, si esprime come:

$$\begin{cases} \max \sum_{i=1}^{n} c_i x_i \\ \sum_{i=1}^{n} a_i x_i \leq b \\ \sum_{i=1}^{n} p_i x_i \leq p \\ ... \\ x_i \in \{0, 1\} \end{cases}$$

In forma non binaria, si sostituisce come prima il vincolo $x_i \in \{0, 1\}$ con $x_i \geq 0$.

In generale, questo tipo di problema non è semplice da risolvere, anche attraverso metodi euristici. In questo corso ci si limita a riportarne l'esistenza, e non vengono dati algoritmi risolutivi.

19.2 Riassunto sugli algoritmi di valutazione

Prendiamo in considerazione il poliedro generico di ILP:

$$\begin{cases} Ax \leq b \\ x \in \mathbb{Z}^n \end{cases}$$

Facciamo un riassunto sulle tecniche greedy per il ricavo di una soluzione ammissibile (che danno un limite inferiore v_I) e sul ricavo del rilassato (che dà un limite superiore v_S), con riferimento alla ILP. Bisogna notare che un algoritmo greedy da vincolo inferiore su problemi di massimo, e superiore altrimenti. Allo stesso modo, il rilassato dà vincolo superiore su problemi di massimo, e inferiore altrimenti.

19.2.1 Algoritmi per valutazioni inferiori

- **Zaino:** abbiamo visto gli **algoritmi dei rendimenti** per il caricamento di problemi di zaino in ILP, cioè per uno zaino di vettore valore v e peso p, con peso massimo P, si calcolano i rendimenti:
$$r_i = \frac{v_i}{p_i}$$
e si selezionano uno o più elementi di rendimento massimo (dipende anche dal tipo booleano o meno di problema), saturando fino a frazione se si è nel rilassato, e fino a intero in caso contrario;

- **Bin-packing:** per la ricerca di soluzioni ammissibili dei problemi di bin-packing abbiamo visto tre algoritmi di ricerca, **next-fit decreasing**, **first-fit decreasing** e **best-fit decreasing**;

- **Algoritmo delle toppe:** l'algoritmo delle toppe per i problemi TSP asimmetrici rappresenta ancora un metodo euristico per il calcolo di cicli hamiltoniani a costo minimo.

19.2.2 Algoritmi per valutazioni superiori

Abbiamo introdotto il concetto di rilassamento di vincoli, in modo da trovare un sovrainsieme della regione ammissibile di ILP da dove è più facile ricavare una soluzione ammissibile. In particolare, la tecnica più immediata è quella di rimuovere il vincolo di interezza ($x \in \mathbb{Z}^n$) o booleano ($x \in \{0,1\}^n$).

19.3 TSP simmetrico

Veniamo quindi ai problemi di TSP simmetrico, cioè dove la tabella dei costi è simmetrica, ovvero in forma:

$$c = \begin{pmatrix} 27 & 23 & 24 & 26 \\ - & 21 & 32 & 33 \\ - & - & 41 & 42 \\ - & - & - & 47 \end{pmatrix}$$

dove notiamo si mettono in relazione i nodi da 1 a $n-1$ sulle righe con i nodi da 2 a n sulle colonne (per cui la diagonale è riempita).

Siccome questo è un caso particolare del TSP, possiamo effettivamente usare tutti i metodi studiati per il TSP asimmetrico. Vediamo però che ci sono delle semplificazioni particolari che possiamo fare sul problema.

Innanzitutto, visto che possiamo prendere gli archi come non orientati, le variabili possibili sono dimezzano. Se nello scorso problema avevamo $5 \cdot 5 = 25$ variabili meno 5 della diagonale, ergo $n^2 - n$, adesso ne abbiamo soltanto $\frac{n^2-n}{2}$.

Abbiamo poi che i vincoli si presentano in forma:

$$\begin{cases} \min c^\intercal x \\ \sum\limits_{h,i \in A} x_{hi} + \sum\limits_{i,k \in A} = 2, \quad \forall i \in N \\ \sum\limits_{i \in S,\, j \notin S} x_{ij} + \sum\limits_{i \notin S,\, j \in S} x_{ij} \geq 1, \quad \forall S \subset N, \quad 2 \leq |S| \leq \left\lceil \frac{|N|}{2} \right\rceil \end{cases}$$

La prima serie di vincoli rappresenta il fatto che ogni nodo è parte di al massimo due archi, ergo uno uscente e entrante (anche se questa definizione non ha molto significato per grafi non orientati). La seconda serie rappresenta invece i vincoli di connessione, che in questo caso prendiamo in entrambe le direzioni, e che limitiamo in dimensioni di sottoinsieme alla cardinalità $|N|$ fratto 2, al limite approssimata all'eccesso, in quanto il vincolo per S vale anche per $N \setminus S$. Anche in questo caso si evitano gli insiemi singoletti con $|S| = 1$, in quanto rappresentano i vincoli già espressi nella seconda riga.

19.3.1 Valutazioni inferiori e superiori

Un'approccio greedy alla soluzione, che fornisce un vincolo superiore v_S (siamo in minimo), è quello di scegliere un nodo ad arbitrio e proseguire da lì in poi scegliendo il nodo con $\min(c_{xj})$ di adiacenza, ovvero:

Algoritmo 9 del nodo vicino

Input: un insieme N di nodi
Output: una valutazione superiore v_S del TSP simmetrico
Parti da un nodo ad arbitrio;
ciclo:
Scegli il nodo adiacente più vicino e unifica
if ci sono altri nodi **then**
  Torna a ciclo
end if

Per il calcolo di una valutazione inferiore avremo bisogno di un rilassamento. Scegliamo di rilassare i vincoli sul grado di tutti i nodi tranne uno, detto k:

$$\begin{cases} \min c^\intercal x \\ \sum_{h,r \in A} x_{hi} + \sum_{r,k \in A} = 2, \quad \forall i \in N \\ \sum_{i \in S,\, j \notin S} x_{ij} + \sum_{i \notin S,\, j \in S} x_{ij} \geq 1, \quad \forall S \subset N \setminus \{r\}, \quad 2 \leq |S| \leq \left\lceil \frac{|N|}{2} \right\rceil \end{cases}$$

Diamo quindi la seguente definizione:

Definizione 19.1: K-albero

Scelto un nodo k, chiamiamo k-albero un insieme di n archi dove $n-2$ archi formano un albero di copertura sul sottografo formato dai nodi $N \setminus \{k\}$, e 2 archi incidono sul nodo k.

In sostanza, un k-albero è un **albero di copertura** con un ciclo. Bisogna notare che questa definizione di k-albero differisce da quella comunemente usata sia in informatica (albero con ramificazione k) che in teoria dei grafi (un altro tipo di grafo non diretto). Inoltre, si trova anche sotto altri nomi, tra cui r-albero, che però è ancora conflitto con gli r-grafi informatici (che sono un tipo di albero per l'organizzazione di informazione spaziale).

Tolta quest'ultima precisazione, abbiamo che ciò che vogliamo è un k-albero minimo. Ricordiamo quindi il seguente algoritmo:

19.3.2 Algoritmo di Kruskal

Possiamo usare l'**algoritmo di Kruskal** per trovare l'albero di copertura minimale per un insieme N di nodi. Assumendo grafi connessi, si può formulare come:

Algoritmo 10 di Kruskal

Input: un insieme N di nodi
Output: un albero di copertura minimale di N
ciclo:
Ordina gli archi in maniera crescente rispetto al peso
Scegli il primo arco.
if aggiungere l'arco non crea cicli **then**
  Aggiungi l'arco e unifica le componenti
else
  Scegli il prossimo arco e riprova
end if
if non hai finito gli archi **then**
  Vai a ciclo
end if

Questo algoritmo trova sempre una soluzione, cioè un albero di copertura minimale, e visto che sceglie sempre archi a costo minimo, trova sempre la soluzione ottima.

Ritornando al problema dei k-alberi minimi, possiamo finalmente presentare il seguente algoritmo per il calcolo di un k-albero di costo minimo su qualsiasi nodo k:

Algoritmo 11 del k-albero

Input: un insieme N nodi, di cui $k \in N$
Output: un k-albero di costo minimo
Trova l'albero di copertura minimale C_H con Kruskal del sottografo $N \setminus \{k\}$
Connetti k a C_H attraverso i due nodi a costo minimo

Possiamo quindi dire perchè abbiamo introdotto il concetto di k-albero: il problema rilassato posto prima, dove si erano rilassate le cardinalità degli archi su tutti i nodi tranne k, ha come soluzione il k-albero a costo minimo. Trovando questo k-albero, abbiamo una valutazione superiore del problema di TSP simmetrico di partenza.

20 Lezione del 24-10-24

20.0.1 Riassunto TSP simmetrico

Avevamo quindi posto problemi con vincoli in forma:

$$\begin{cases} \min c^\intercal \cdot x \\ \sum\limits_{x<j} x_{ij} + \sum\limits_{j<y} x_{ij} = 2, \quad \forall j \\ \sum\limits_{i \in S\, j \notin S} x_{ij} + \sum\limits_{i \notin S\, j \in S} x_{ij} \geq 1, \quad \forall S \subset N, \quad 2 \leq |S| \leq \left\lceil \frac{|N|}{2} \right\rceil \end{cases}$$

cioè dove si poneva la somma dei nodi entranti e uscenti (i vincoli *di grado*) da j come $= 2$. Visto che j era l'unico nodo su cui era imposto il vincolo, si aveva che la soluzione del problema era il j-albero a costo minimo.

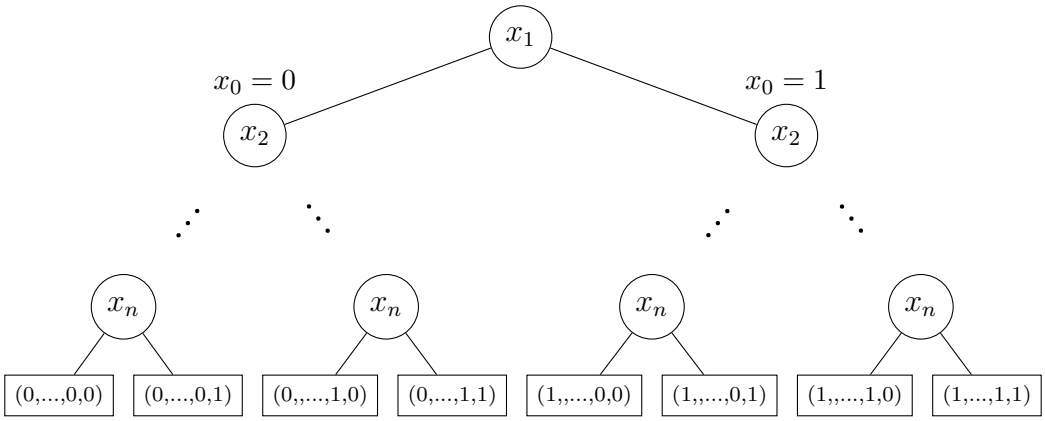
Inoltre, si aveva che i vincoli sulla terza erano quelli *di connessione*, che si cercavano solo su cardinalità dei sottoinsiemi $= \left\lceil \frac{|N|}{2} \right\rceil$, in quanto i vincoli per S valevano anche per $N \setminus S$.

20.1 Branch and bound

Avevamo visto l'algoritmo dei piani di taglio di Gomory per il calcolo di soluzioni approssimate di problemi ILP. Presentiamo adesso un altro metodo, detto **branch and bound**. Il metodo più naive che possiamo adottare per risolvere un problema in forma:

$$\begin{cases} \max c^\intercal x \\ Ax \leq b \\ x \in \{0,1\}^n \end{cases}$$

è quello di enumerare tutti le possibili soluzioni ammissibili $\in \{0,1\}^n$, costruendo il cosiddetto **albero di enumerazione**. Scegliamo quindi una variabile, x_1, e costruiamo l'albero:



Abbiamo che il calcolo di ogni nodo dell'albero richiede 2^n operazioni (nodi di un albero binario). Questo calcolo, però, non è effettivamente necessario nella maggior parte dei casi. Chiamiamo quindi **problemi** P_{ij} ogni nodo dell albero, con i che seleziona il livello e j il fratello, da sinstra verso destra.

Stabilita una valutazione superiore e inferiore di ogni problema, che chiameremo $v_S(P)$ e $v_I(P)$, abbiamo:

$$v_I(P) \leq v(P) \leq v_S(P)$$

Il funzionamento dell'algoritmo del branch and bound è determinato proprio dall'esistenza di valutazioni "buone" inferiori e superiori. Possiamo infatti usare queste valutazioni per stabilire **regole di taglio** che ci permettano di tagliare (si dice anche *visitare implicitamente*) un'intero sottoalbero a partire da un certo nodo P_{ij}. Queste regole di taglio assicurano, essenzialmente, che ogni sottoproblema $P_{i+k,j'}$ figlio di P_{ij} non contiene l'ottimo, e quindi si può saltare.

Vediamo quindi le regole di taglio che possiamo adottare. Mantendendo un'ottimo corrente x, abbiamo che calcolato un nuovo P_{ij} per enumerazione:

1. $P_{ij} = \emptyset$ significa che per un certo nodo P_{ij} possiamo tagliare tutti i problemi che istanziano le i variabili di P_{ij}, ergo tutti i suoi nodi figli;

2. Calcolo di $v_S(P_{ij})$. Se $v_S(P_{ij}) < v_I(P)$ del problema, allora posso scartare il sottoalbero: non troverò soluzioni migliori scendendovi;

3. $v_S(P_{ij}) > v_I(P)$, e l'$\bar{x}$ dove si ha tale v_S è ammissibile per P, allora si prende $\bar{x}$ come nuovo x e si fa una visita implicità di P_{ij}: questo perchè un suo sottoproblema non potrà darci di meglio (più scendiamo nell'albero, più stringiamo i vincoli, ergo $P_{i+k,j'}$ figlio di P_{ij} ha $v_S(P_{i+k,j'}) \leq v_S(P_{ij})$).

Ricordiamo che quanto detto finora vale su problemi **massimizzanti**: su problemi **minimizzanti** dovremo invertire l'ordine delle disugaglianze, e prendere vincoli superiori anziché inferiori e viceversa.

21 Lezione del 29-10-24

21.1 Regole di branch

Vediamo quindi nel dettaglio le regole di ramificazione da usare nell'applicazione del branch and bound, su problemi di minimo e di massimo. Consideriamo un problema TSP simmetrico minimizzante:

21.1.1 Regole di branch minimizzanti

Partiamo dal problema P, e istanziamo variabili per ottenere una serie di $P_{i,j}$, dove la i rappresenta il livello di profondità nell'albero e j il sottoproblema corrente a quel livello.

Ogni volta che si istanzia la prima cosa da controllare è l'ammissibilità delle soluzioni ottenute istanziando le variabili. Se si ha infatti che se $P_{i,j} = \emptyset$, quel sottoalbero può essere tagliato.

Si controlla poi se vale $v_I(P_{i,j}) \geq v_S(P)$, cioè se il minimo possibile ottenuto dal sottoproblema $P_{i,j}$ è più piccolo della valutazione superiore del problema P. In questo caso si taglia il sottoalbero.

Si controlla infine se vale $v_I(P_{i,j}) < v_S(P)$ e questa v_I è ammissibile per P. Se sì, allora si aggionra il v_S corrente di P a v_I. Inoltre, si può applicare anche la regola precedente, e quindi tagliare il sottoalbero.

Riassumendo si hanno quindi le regole:

- $P_{i,j} = \emptyset \implies$ taglio;
- $v_I(P_{i,j}) \geq v_S(P) \implies$ taglio;
- $v_I(P_{i,j}) < v_S(P)$ e $v_I \in P \implies v_S(P) \leftarrow v_I(P_{i,j})$, taglio.

Consideriamo poi un problema di zaino booleano massimizzante:

21.1.2 Regole di branch massimizzanti

In un problema massimizzante si segue un approccio simile. Dal problema P si istanzia successivamente per ottenere $P_{i,j}$ con le stesse caratteristiche di prima.

Istanziando, se si va a svuotare la regione ammissbilie, cioè si ottiene $P_{i,j} = \emptyset$, si scarta quel sottoalbero.

Si controlla poi se vale $v_S(P_{i,j}) \leq v_I(P)$, cioè se il massimo possibile ottenuto dal sottoproblema $P_{i,j}$ è più grande della valutazione inferiore del problema P. In questo caso, chiaramente, si taglia il sottoalbero.

Infine si controlla se vale $v_S(P_{i,j}) > v_I(P)$, e questa v_S è ammissibile per P (cioè, in questo caso, è a componenti intere). Se sì, allora si aggiorna il v_I corrente di P a v_S. Inoltre, si taglia il sottoalbero.

Anche qui, riassumendo, si hanno le regole:

- $P_{i,j} = \emptyset \implies$ taglio;
- $v_S(P_{i,j}) \leq v_I(P) \implies$ taglio;
- $v_S(P_{i,j}) > v_I(P)$ e $v_S \in P \implies v_I(P) \leftarrow v_S(P_{i,j})$, taglio.

22 Lezione del 30-10-24

22.1 Problema di copertura

Problema 22.1: di copertura

Supponiamo che l'ASL debba dislocare ambulanze su 5 sedi distribuite sul territorio, in 9 diverse località. La matrice di adiacenza fra sedi e località è la seguente:

	Sede 1	**Sede 2**	**Sede 3**	**Sede 4**	**Sede 5**
Località 1	1	0	1	0	1
Località 2	0	1	0	1	0
Località 3	1	1	0	0	0
Località 4	0	1	1	0	1
Località 5	0	0	0	0	1
Località 6	1	1	1	1	1
Località 7	1	1	0	0	1
Località 8	0	1	1	1	0
Località 9	0	0	1	0	1

Vogliamo capire in quali sedi dovremo dislocare le ambulanze in modo da coprire tutte le località.

Il problema è risolvibile attraverso la ILP. Possiamo assumere che la matrice di adiacenza sia stata ricavata da qualche matrice dei tempi (rappresentante il tempo necessario a raggiungere una località), e quindi tagliata su un tempo massimo di, ad esempio, 20 minuti, con il risultato che le località a meno di 20 minuti di distanza risultano connesse e quelle a più di 20 minuti disconnesse.

Possiamo quindi definire alcune **regole di riduzione** per semplificare la matrice di adiacenza:

1. Le righe di soli zeri $(0, ..., 0)$ possono essere eliminate in quanto rappresentano una soluzione impossibile del problema;

2. Le righe di soli uni $(1, ..., 1)$ possono essere eliminate in quanto possono essere risolte da qualsiasi servizio;

3. Se una riga contiene un solo 1, apro un "servizio" per quella riga (cioè gli dedico una delle colonne), la elimino, e elimino tutte le righe che hanno 1 sulla stessa colonna.

4. Se per due colonne r e k, $r_{ij} \geq k_{kj}$ per ogni riga, allora si diche che r **domina** k, e quindi che k può essere scartata. Questo vale solo se l'apertura è a costo costante su tutte le colonne: in caso contrario potremmo scartare opzioni viabili di ottimizzazione.

Vogliamo capire quali località dovranno essere servite da quale sede in modo da coprirle tutte. Si ricavano quindi i vincoli su ogni riga:

$$\begin{cases} x_1 + x_3 + x_5 \geq 1 \\ x_2 + x_4 \geq 1 \\ ... \\ x_3 + x_5 \geq 1 \\ x_i \in \{0, 1\} \end{cases}$$

ergo si ricava un problema di ILP in forma:

$$\begin{cases} \min c^\intercal \cdot x \\ Ax \geq 1 \\ x \in \{0, 1\} \end{cases}$$

dove c rappresenta un vettore costo, nel problema riportato assunto come costante.

Quello che facciamo effettivamente è scegliere fra n sottoinsiemi, scegliendo l'insieme minimo di questi per coprire l'interezza degli elementi (detti anche nodi).

22.1.1 Valutazioni inferiori e superiori

Vediamo quali valutazioni possiamo usare per trovare soluzioni approssimate.

- **Valutazione inferiore:** una valutazione inferiore v_I può essere agevolmente calcolata dal rilassato continuo del problema ILP:

$$\begin{cases} \min c^\intercal \cdot x \\ Ax \geq 1 \\ 0 \leq x \leq 1 \end{cases}$$

- **Valutazione superiore:** si può applicare un algoritmo greedy per il calcolo di una valutazione superiore, assunti costi costanti:

Algoritmo 12 di copertura a costi costanti

Input: un problema di copertura
Output: una valutazione superiore v_S
ciclo:
$r_i \leftarrow$ la somma dei valori su ogni colonna
Rimuovi la colonna con r_i maggiore, scegli quella colonna nella soluzione, e rimuovi le righe servite da quella colonna
Torna a ciclo

Notiamo che r_i rappresenta effettivamente un rendimento, come avevamo visto nei problemi di zaino. Possiamo infatti applicare un'altro algoritmo, più sofisticato, nel caso dei costi variabili:

Algoritmo 13 di Chvatal

Input: un problema di copertura
Output: una valutazione superiore v_S
ciclo:
$r_i \leftarrow$ il rapporto fra il costo di ogni colonna e la somma dei valori su quella colonna
Rimuovi la colonna con r_i maggiore, scegli quella colonna nella soluzione, e rimuovi le righe servite da quella colonna
Torna a ciclo

22.1.2 Problemi di massima copertura

Supponiamo di avere una stima del valore (che possiamo assimilare alla domanda, o al guadagno associato all'apertura di un servizio su un certo nodo) su ogni riga (nell'esempio precedente di ogni località), e di voler prediligere soluzioni che coprono righe con valore maggiore, supponendo di avere un numero limitato di k risorse (in questo caso ambulanze) a disposizione. Prendiamo ad esempio la tabella:

	Abitanti	**Sede 1**	**Sede 2**	**Sede 3**	**Sede 4**	**Sede 5**
Località 1	2000	1	0	1	0	1
Località 2	1000	0	1	0	1	0
Località 3	800	1	1	0	0	0
Località 4	3000	0	1	1	0	1
Località 5	2500	0	0	0	0	1
Località 6	1200	1	1	1	1	1
Località 7	1700	1	1	0	0	1
Località 8	400	0	1	1	1	0
Località 9	150	0	0	1	0	1

Il problema potrebbe essere di assegnare ambulanze alle sedi che offrono copertura della maggior popolazione possibile. Formuleremo allora il problema di ILP:

$$\begin{cases} \max h_1 z_1 + ... + h_9 z_9 \\ x_1 + x_3 + x_5 \geq z_1 \\ x_2 + x_4 \geq z_2 \\ ... \\ x_3 + x_5 \geq z_9 \\ x_1 + x_2 + x_3 + x_4 = k \\ x \in \{0, 1\} \\ z \in \{0, 1\} \end{cases}$$

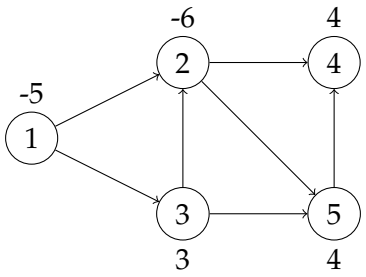
Questo rappresenta un **problema di massima copertura**. I coefficienti $h_1, ..., h_n$ sono i valori associati ad ogni nodo (qui ad ogni località): la funzione obiettivo risponde meglio all'apertura dei servizi ai nodi con valore alto. I successivi n vincoli rappresentano il fatto che, presa un nodo z, cioè una riga, dobbiamo rendere uguale a 1 almeno una fra le colonne che la servono, cioè dobbiamo aprirgli almeno un servizio. Infine, il vincolo $x_1 + ... + x_n = k$ rappresenta il fatto che vogliamo prendere k servizi, cioè colonne (nel nostro caso distribuire k ambulanze).

Quello che facciamo, effettivamente, è scegliere fra i sottoinsiemi x per massimizzare la copertura sugli elementi z, massimizzando il guadagno ottenuto in base alla scelta degli z (scelti gli z i successivi vincoli determinano quali x dobbiamo aprire per servirli).

23 Lezione del 05-11-24

23.1 Ottimizzazione sui grafi

Veniamo adesso all'**ottimizzazione su grafi**, cioè la massimizzazione o minimizzazione del flusso su grafi definiti come insiemi $G = (N, A)$, di $\text{card}(N) = n$ nodi e $\text{card}(A) = m$ archi, dove $A \subseteq N \times N$ (dalla commutatività del prodotto cartesiano si parla di grafi **orientati**). Si potrà avere ad esempio:



Etichettiamo con il **bilancio** di flusso ogni nodo, adottando la convenzione dei **segni positivi** per i *pozzi* (i nodi che assorbono flusso) e **segni negativi** per le *sorgenti* (i nodi che erogano flusso). A questo punto, il vettore soluzione x_{ij} rappresenterà le unità di flusso su ogni arco (i e j sono indici rispettivamente dei nodi di partenza e arrivo). Come sempre, il vettore soluzione è ordinato **lessicograficamente**.

Quello che vogliamo fare è quindi esprimere i **vincoli** a cui è sottoposto flusso sugli archi del grafo. Conviene quindi guardare sempre al bilancio in termini di flusso di ogni nodo, prendendo gli archi entranti sul nodo (cioè che partono da nodi che *approvigionano* il nodo) come positivi e gli archi uscenti dal nodo (cioè che vanno a nodi che si *riforniscono* dal nodo) come negativi (da qui il segno negativo alle sorgenti). Ad esempio, avremo per l'esempio precedente:

$$\begin{cases} -x_{12} - x_{13} = -5 \\ x_{12} + x_{32} - x_{24} - x_{25} = -6 \\ x_{13} - x_{32} - x_{35} = 3 \\ x_{24} + x_{54} = 4 \\ x_{25} + x_{35} - x_{54} = -4 \\ x_{ij} \geq 0 \end{cases}$$

in formato duale standard.

La situazione diventa un problema di LP quando si introduce un **costo di movimentazione** su ogni arco: a questo punto può essere interessante calcolare, se ogni unità di flusso "paga" il costo dell'arco su cui scorre, il **flusso di costo minimo** sul grafo. Questo si traduce nella funzione costo:

$$\min 2x_{12} + 3x_{13} + 4x_{24} + 6x_{25} + 7x_{32} + 5x_{35} + 8x_{54} \equiv c^{\intercal} \cdot x$$

e quindi dà il sistema completo:

$$\begin{cases} \min 2x_{12} + 3x_{13} + 4x_{24} + 6x_{25} + 7x_{32} + 5x_{35} + 8x_{54} \\ -x_{12} - x_{13} = -5 \\ x_{12} + x_{32} - x_{24} - x_{25} = -6 \\ x_{13} - x_{32} - x_{35} = 3 \\ x_{24} + x_{54} = 4 \\ x_{25} + x_{35} - x_{54} = -4 \\ x_{ij} \geq 0 \end{cases}$$

23.1.1 Matrici di incidenza

Diamo una definizione più generale dell'insieme di vincoli necessari a definire un problema di ottimizzazione su un grafo. Avevamo che vogliamo, preso il vettore di bilanci b_i, eguagliare il bilancio del flusso effettivo su un nodo in funzione degli archi entranti ed uscenti con questo vettore. Questo si esprime nella forma:

$$\sum_{(e,i)\in A} x_{ei} - \sum_{(i,u)\in A} x_{iu} = b_i$$

dove si è preso con A l'insieme di tutti gli archi, con $(e,i) \in A$ gli indici dei nodi che formano (*incidono su*) archi entranti al nodo, e con $(i,q) \in A$ gli indici dei nodi che formano archi uscenti dal nodo, cioè la definizione operativa di bilancio del flusso che avevamo usato nell'esempio. Più formalmente, abbiamo che i vincoli di un problema di ottimizzazione su un grafo rispettano la forma:

$$Ex = b$$

dove E è la **matrice di incidenza** del grafo.

Definizione 23.1: Matrice di incidenza

Per un grafo di $\text{card}(N) = n$ nodi e $\text{card}(A) = m$ archi, la matrice di incidenza E è la matrice $n \times m$ che mette in relazione ogni nodo con ogni arco, con i valori e_{ij}:

- $e_{ij} = 0$: il nodo non partecipa all'arco;
- $e_{ij} = -1$: il nodo è la partenza dell'arco;
- $e_{ij} = 1$: il nodo è l'arrivo dell'arco.

Riassumiamo la differenza fra **matrice di incidenza** e **matrice di adiacenza** (che abbiamo già usato, ad esempio nei problemi di assegnamento di costo minimo e di trasporto). La prima mette in relazione nodi con archi, la seconda nodi con nodi. Entrambe sono utili alla modellizzazione di problemi di tipo diverso. A scopo esplicativo, si riportano le matrici di incidenza E e di adiacenza A dell'esempio precedente:

$$E = \begin{array}{c|ccccccc} & (1,2) & (1,3) & (2,4) & (2,5) & (3,2) & (3,5) & (5,4) \\ \hline 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & -1 & -1 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & -1 & -1 & 0 \\ 4 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 5 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \end{array} \qquad A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Abbiamo quindi che la forma più generale di un problema di ottimizzazione su un grafo è:

$$\begin{cases} \min c^\intercal \cdot x \\ Ex = b \\ x \geq 0 \end{cases}$$

dove segnaliamo è importante il grafo sia **connesso**:

Definizione 23.2: Grafo connesso

Un grafo si dice connesso quando l'insieme degli archi A contiene un albero di copertura $T \subseteq A$.

Questa definizione (che in verità si potrebbe intendere come caratterizzazione) si fonda sull'idea di **albero di copertura** (che in verità abbiamo già usato nei problemi del commesso viaggiatore in ILP), che definiamo come segue:

Definizione 23.3: Albero di copertura

Dato un grafo $G = (N, A)$, un albero di copertura è un sottoinsieme di archi $T \subseteq A$ tale che il sottografo (N, T) è connesso e non presenta cicli. Un nodo di questo grafo su cui incide solo un arco si dice **foglia**.

Notiamo che su ogni colonna della matrice di adiacenza si ha esattamente un 1 e un -1, e tutti gli altri elementi valgono 0 (in altre parole, un arco è formato da un nodo di partenza e un nodo di arrivo). Questo significa che la somma delle righe della matrice è nulla (lo è la somma di ogni colonna), ergo per definizione queste sono fra di loro linearmente dipendenti. Si ha quindi che:

Teorema 23.1: Rango della matrice di incidenza

La matrice di incidenza di un problema di ottimizzazione su un grafo con n nodi ha rango $n - 1$.

Questo significa che il sistema $Ex = b$ è **sovradeterminato**, e possiamo quindi tranquillamente rimuoverne una riga (solitamente si sceglie la prima).

La proposizione precedente rispetto alla dipendenza lineare delle righe della matrice di adiacenza dà un limite superiore sul suo rango: mostriamo come dare un limite inferiore (e quindi dimostrare il teorema precedente). Per formare un albero di copertura si prendono i minimi $n - 1$ archi necessari a collegare fra di loro tutti i nodi, ergo si prendono le $n - 1$ colonne della matrice di adiacenza del grafo corrispondenti agli archi che formano l'albero per formare un minore E_T. Da queste, permutando appositamente righe e colonne, si può sempre ottenere una forma del tipo:

$$E_T = \left(\begin{array}{c|c} \pm 1 & 0 \dots 0 \\ \hline * & E_S \end{array}\right)$$

dove la prima riga rappresenta una foglia z, e la matrice E_S la matrice di adiacenza del sottoalbero di copertura ottenuto su $N \setminus z$. Questa forma, per ipotesi induttiva, ci dice che la matrice di incidenza dell'albero è **triangolare inferiore**, da cui ricaviamo due cose:

- La matrice di adiacenza dell'albero di copertura, ergo il minore, è invertibile, da cui il rango $n-1$;
- La matrice di adiacenza dell'albero ha determinante ± 1, ergo è **unimodulare**. Questo ha dei risvolti importanti per quanto riguarda l'ILP, in quanto vedremo che una soluzione di LP per un problema di flusso minimo corrisponde con la soluzione del problema di ILP corrispondente (a patto di bilanci b_i interi).

23.1.2 Caratterizzazione delle basi

Abbiamo appurato che la matrice di incidenza di un problema di ottimizzazione su un grafo ha rango $n-1$ e presenta minori invertibili, rappresentanti matrici di incidenza di alberi di copertura. Visto che il problema stesso era stato espresso in forma duale standard, e che quello che facciamo nel prendere minori di rango $n-1$ è effettivamente scegliere $n-1$ colonne della matrice di incidenza, si ha che ogni albero di copertura rappresenta una **base** del poliedro.

Teorema 23.2: Caratterizzazione di base di matrici di incidenza

Per una matrice di incidenza E di un problema di ottimizzazione su grafi, un albero di copertura coincide con una data base T ottenuta scegliendo le $n-1$ colonne degli archi che formano l'albero, ovvero:

$$T \text{ è un albero di copertura} \Leftrightarrow T \text{ è una base}$$

Vediamo un modo per calcolare una soluzione di base data una base T. Questa prenderà il nome di **flusso di base**. Iniziamo con l'impostare, come era sempre valso per il duale standard, a 0 le variabili non di base. Adesso si può fare una visita positicipata per foglie (considerando la base come un albero). Vorremo che questa foglia rispetti:

$$E_T x_T = b$$

Partendo dalle foglie, si può mettere sull'arco corrispondente (e quindi sulla x_{ij} corrispondente) il bilancio di quella foglia, e poi tagliarla dall'albero. Il procedimento si ripete sulle foglie rimaste, tenendo conto dei bilanci dei nodi rimanenti, fino ad arrivare alla radice dell'albero.

Così facendo troveremo un vettore soluzione x_{ij}, su cui notiamo se almeno una componente è < 0, la soluzione di base non è ammissibile (significa che un nodo sorgente chiede più di quanto può erogare).

Possiamo quindi dare, stabilito un flusso di base, la seguente caratterizzazione di ammissibilità:

Definizione 23.4: Caratterizzazione dei flussi di base ammissibili

Un flusso di base, scelto T albero di copertura, è ammissibile se $\forall (i,j) \in T$ si ha $\overline{x}_{ij} \geq 0$.

e degenrazione, posto L come l'insieme totale degli archi A tolti quelli appartenenti in T, cioè $A \setminus T$:

Definizione 23.5: Caratterizzazione dei flussi di base degeneri

Un flusso di base, scelto T albero di copertura, è degenere se $\exists (i,j) \in L$ per cui $x_{ij} = 0$.

23.1.3 Reti sbilanciate

Notiamo che finora abbiamo assunto l'ipotesi:

$$\sum_{i=1}^{n} b_i = 0$$

Potremmo avere invece che questa sommatoria è *negativa* o *positiva*, ergo il grafo **sbilanciato** in **positivo** o in **negativo**. Nel primo caso, si ha più richeista che produzione, e il problema è quindi vuoto. Nel secondo, si ha puù produzione che richiesta, e bisogna quindi introdurre un **nodo fittizio**, di capacità pari all'**eccedente** $\sum_{i=1}^{n} b_i$, a costo nullo e connesso ad ogni sorgente del grafo, dove verrà a riversarsi (in condizioni di ottimo) il flusso in eccesso.

24 Lezione del 06-11-24

24.1 Dualità nei problemi di ottimizzazione su grafi

Abbiamo visto come l'algoritmo del simplesso si basava sulla dualità fra un problema, espresso in *forma primale standard*, e un suo **duale**, espresso in *forma duale standard*. Abbiamo inoltre detto che l'operazione duale è **involutoria**: il duale del duale è nuovamente il primale, e così via. Si ha che questo è vero anche sui problemi di ottimizzazione su grafi, con l'unica differenza che il problema da cui partiamo è solitamente espresso in *forma duale standard* (come abbiamo visto negli esempi precedenti), e che vi applicheremo l'operazione duale per ricavarne, sì un duale, ma che verrà espresso in *forma primale standard*. Chiameremo comunque **primale** il primo problema, e **duale** il secondo.

24.1.1 Forma della matrice dei vincoli

Facciamo innanzitutto una precisazione sulla forma in cui si esprime la matrice A (o nei problemi su grafi, E di incidenza). Sono equivalenti le forme:

$$Ax = b \Leftrightarrow x^\intercal A^\intercal = b$$

e quindi sui grafi:

$$Ex = b \Leftrightarrow x^\intercal E^\intercal = b$$

La seconda forma ha un vantaggio: la matrice così espressa risulta identica sia nel primale che nel duale, in quanto avevamo visto i vincoli duali si esprimono, almeno nei problemi di ottimizzazione visti finora, come:

$$A^\intercal y = c$$

24.2 Duali di problemi di ottimizzazione sui grafi

Veniamo quindi al calcolo vero e proprio dei problemi associati di problemi di ottimizzazione sui grafi. A partire dal problema:

$$\begin{cases} \min c^\intercal \cdot x \\ Ex = b \\ x \geq 0 \end{cases}$$

potremo ricavare:

$$\begin{cases} \max b^\intercal \cdot \pi \\ \pi^\intercal E \leq c \Leftrightarrow E^\intercal \pi \leq c \end{cases}$$

dove si nota, come prima, l'equivalenza delle due forme di espressione dei vincoli.

Chiamiamo questo problema **problema dei potenziali** ai nodi del grafo. :qa! :qa! Notiamo che, se nel duale avevamo rimosso una riga, dal teorema del rango $n-1$, qui potremo rimuovere una **variabile**, cioè una colonna della matrice $E^\intercal$.

Presa quindi una base T (cioè un albero di copertura), avremo che il **potenziale di base** relativo a T è la soluzione del sistema $\pi^\intercal E_T = c_T^\intercal$, cioè:

$$\pi^\intercal = c^\intercal E_T^{-1}$$

Il potenziale di base, come il flusso di base, ha un algoritmo greedy di calcolo piuttosto intuitivo: partendo dalla **radice** dell'albero T, si calcola il costo complessivo sugli archi necessario a spostarsi da questa al nodo di cui vogliamo trovare il potenziale.

Dopo il calcolo dei potenziali, potremo calcolare i cosiddetti **costi ridotti**, come la differenza fra il costo di ogni arco e la **differenza di potenziale** dei nodi che collega:

$$c_{ij}^\pi = c_{ij} + \pi_i - \pi_j$$

Notiamo come l'unico valore che ci interessa riguardo ai potenziali non sono i potenziali stessi, ma le loro differenze.

Una volta stabilito il vettore dei potenziali, e quindi dei costi ridotti (sul duale), si può dare la seguente caratterizzazione di vertici ammissibili:

Definizione 24.1: Caratterizzazione dei potenziali di base ammissibili

Un potenziale di base, posto T albero di copertura, è ammissibile se $\forall (i,j) \in L$ si ha $c_{ij}^\pi \geq 0$.

e degeneri, ricordando l'ortogonalità di queste due caratteristiche:

Definizione 24.2: Caratterizzazione dei potenziali di base degeneri

Un potenziale di base, posto T albero di copertura, è degenere se $\exists (i,j) \in L$ tale per cui $c_{ij}^\pi = 0$.

Infine, possiamo riformulare il concetto di **scarti complementari** per soluzioni di base complementari nei problemi di ottimizzazione su grafi:

- Un flusso ammissibile x è ottimo se e solo se esiste un potenziale π tale che:

$$\begin{cases} x_{ij} = 0 \implies c_{ij}^\pi \geq 0 \\ x_{ij} > 0 \implies c_{ij}^\pi = 0 \end{cases}$$

- Un potenziale ammissibile π è ottimo se e solo se esiste un flusso x tale che:

$$\begin{cases} c_{ij}^{\pi} = 0 \implies x_{ij} \geq 0 \\ c_{ij}^{\pi} > 0 \implies x_{ij} = 0 \end{cases}$$

Applicando la dualità forte e le condizioni di ammissibilità riportate prima, possiamo formulare il seguente teorema:

Teorema 24.1: di Bellman

Una partizione (T, L) di archi con T albero di copertura che genera un flusso di base x ammissibile è ottima se:

$$c_{i,j}^{\pi} \geq 0 \quad \forall (i,j) \in L$$

Questo teorema non rivela altro che un risultato già noto: se una soluzione è ammissibile sia nel primale che nel duale, allora è ottima.

25 Lezione del 07-11-24

25.1 Simplesso per flussi

Vediamo come applicare l'algoritmo del simplesso ai problemi di ottimizzazione sui grafi, in particolare per risolvere problemi di flusso minimo.

Partiamo dall'assunto fondamentale che una soluzione ammissibile è rappresentata da un albero di copertura che genera flusso ammissibile, che chiameremo **albero di copertura ammissibile**. Potremo quindi partizionare l'insieme A degli archi in due sottoinsiemi:

- T, formato dagli archi che formano l'albero (e che denota quindi l'albero stesso);
- L, formato dagli archi rimanenti.

L'insieme di possibili partizioni (T, L) rappresenta le basi dei vertici di un poliedro, cioè qualsiasi albero di copertura ammissibile T rappresenta una base del poliedro.

Si possono quindi calcolare, dato T, i costi ridotti c_{ij}^{π} su tutti gli archi che comprende:

$$c_{ij}^{\pi} = c_{ij} + \pi_i - \pi_j$$

Si ha che, dal teorema di Bellman, se $\forall (i,j) \in L : c_{ij}^{\pi} \geq 0$, allora la base duale è ammissibile e siamo all'ottimo. Altrimenti, dovrà essere che $\exists (i,j) \in L : c_{ij}^{\pi} < 0$. Scegliamo questo (i,j) come **arco entrante**.

Si ha che l'arco entrante (i,j) forma un ciclo con gli archi dell'albero T: la selezione di un arco uscente coinciderà nel simplesso su grafi ad un processo di *eliminazione di cicli*. Si sceglie allora una direzione di percorrenza del ciclo concorde a (i,j), e si partizionano gli archi del ciclo in $\mathcal{C}^+$ per gli archi concordi a questa direzione, e $\mathcal{C}^-$ per gli archi discordi.

L'obiettivo è quello di "spingere" più unità di flusso ϑ possibili nella direzione del ciclo, sfruttando l'arco appena introdotto. Riguardo ai flussi, questo significherà aggiungere ϑ unità di flusso agli archi in $\mathcal{C}^+$, e rimuovere ϑ unità di flusso dagli archi in $\mathcal{C}^-$. Se $\mathcal{C}^-$ è vuoto, non c'è limite al flusso che possiamo spingere, e l'ottimo è $-\infty$. Altrimenti, il valore massimo di ϑ sarà quello che non viola i vincoli, portando un arco in $\mathcal{C}^-$ a valori

< 0. Questo valore non sarà altro che il flusso minimo già impegnato sugli archi x_{ij} con $(i,j) \in C^-$, cioè il flusso dell'arco che si "svuoterà" per primo spingendo flusso nella direzione opposta. Chiamiamo questo arco **arco uscente**.

Si aggiorna quindi la base rimuovendo l'arco uscente e introducendo l'arco entrante, e si ripete fino all'ottimo.

Possiamo quindi formulare l'algoritmo per intero:

Algoritmo 14 del simplesso per flussi

Input: un problema di flusso di costo minimo
Output: la soluzione ottima
Trova una partizione degli archi (T, L) con T albero di copertura che genera un flusso ammissibile
ciclo:
Calcola il flusso di base $\bar{x}_T = E_T^{-1} b$, posti gli $\bar{x}_L = 0$, e il potenziale di base $\bar{\pi}_T = c_T^\intercal E_T^{-1}$

Calcola i costi ridotti $c_{ij}^{\bar{\pi}} = c_{ij} + \bar{\pi}_{ij} - \bar{\pi}_{ij}$ per ogni arco
if $c_{ij}^{\bar{\pi}} \geq 0 \ \ \forall (i,j) \in L$ **then**
  Fermati, $\bar{x}$ è un flusso ottimo e $\bar{\pi}$ è un potenziale ottimo
else
  Calcola l'arco entrante:

$$(p,q) = \min \left\{ (i,j) \in L : c_{ij}^{\bar{\pi}} < 0 \right\}$$

  stabilito l'ordinamento lessicografico degli archi
  Chiama $\mathcal{C}$ il ciclo che l'arco (p,q) forma con gli archi in T
  Fissa un orientamento concorde a (p,q) su $\mathcal{C}$ e partiziona $\mathcal{C}$ in $\mathcal{C}^+$ archi concordi e $\mathcal{C}^-$ archi discordi a tale orientamento
end if
if $\mathcal{C}^- = \emptyset$ **then**
  Fermati, il flusso di costo minimo ha valore $-\infty$
else
  Calcola:

$$\vartheta = \min\{\bar{x}_{ij} : (i,j) \in \mathcal{C}^-\}$$

  e trova l'arco uscente:

$$(r,s) = \min\{(i,j) \in \mathcal{C}^- : \bar{x}_{ij} = \vartheta\}$$

  stabilito un ordinamento lessicografico degli archi
end if
Aggiorna le partizioni come:

$$T = T \setminus \{(r,s)\} \cup \{(p,q)\} \quad L = L \setminus \{(p,q)\} \cup \{(r,s)\}$$

Torna a ciclo

25.1.1 Ottimizzazioni del simplesso per i flussi

Possiamo fare alcune ottimizzazioni considerevoli sull'algoritmo del simplesso per i flussi:

- Non è necessario calcolare i costi ridotti c_{ij}^{π} per ogni arco $\in A$, ma soltanto in L, in quanto sappiamo gli altri (quelli $\in T$) varranno 0;
- Non è necessario calcolare l'insieme $\mathcal{C}^*$: le rimozioni potranno essere effettuate solo da $\mathcal{C}^-$;
- Il calcolo dei potenziali di base dopo l'aggiornamento delle partizioni può essere effettuato senza inversioni di matrici. Si ha che, presi l'arco entrante (p, q) e l'arco uscente (r, s), la rimozione di (r, s) (prima dell'introduzione di (p, q)) partiziona effettivamente i nodi del grafo in due sottoinsiemi N_p e N_q, con $p \in N_p$ e $q \in N_q$. Avremo allora che i nuovi potenziali di base π_i' saranno:

$$\pi_i' = \begin{cases} \pi_i, & i \in N_p \\ \pi_i' + c_{pq}^{\pi}, & i \in N_q \end{cases}$$

  cioè la partizione p resta invariata, e la partizione q sale di potenziale pari al costo ridotto c_{pq}^{π}.

- Una volta variati i potenziali, si possono ricalcolare anche i costi ridotti $c_{ij}^{\pi'}$ con una regola simile:

$$c_{ij}^{\pi'} = \begin{cases} c_{ij}^{\pi'}, & (i,j) \in N_p \vee (i,j) \in N_q \\ c_{ij}^{\pi'} - c_{pq}^{\pi}, & (i,j) \in N_p \wedge (i,j) \in N_q \\ c_{ij}^{\pi'} + c_{pq}^{\pi}, & (i,j) \in N_q \wedge (i,j) \in N_p \end{cases}$$

  cioè, se l'arco è interamente contenuto in N_p o N_q, il costo ridotto resta invariato. Altrimenti, se l'arco "salta" dalla partizione p alla partizione q, si aggiunge o si sottrae, a seconda della direzione, il costo ridotto di $c_p q^{\pi}$.

25.1.2 Ricavare le variazioni dei cambi di partizione

Potremmo voler ricavare le variazioni in termini di valore ottimo che si ottengono dopo un passo del simplesso sui flussi, quindi dopo l'aggiornamento delle partizioni. Abbiamo che la variazione del valore ottimo è quindi il flusso introdotto sul ciclo, cioè ϑ, moltiplicato per il costo ridotto dell'arco introdotto, cioè il valore si evolve di v in v' come:

$$v' = V - \vartheta \cdot c_{ij}^{\pi}$$

26 Lezione del 11-11-24

26.1 Problemi esprimibili come flussi minimi

Diversi problemi che abbiamo già visto possono essere formulati come problemi di flusso minimo su grafi. In particolare, vediamo l'**assegnamento di costo minimo** e il problema di **trasporto**.

26.1.1 Trasporto

Avevamo definito un problema di trasporto come un problema LP in forma:

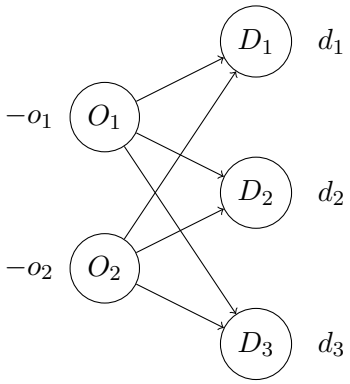
$$\begin{cases} \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \sum_{i=1}^{m} x_{ij} \geq d_j, \quad \forall i = 1, ..., n \\ \sum_{j=1}^{n} x_{ij} \leq o_{ij}, \quad \forall j = 1, ..., m \\ x_{ij} \geq 0 \end{cases}$$

Possiamo concettualizzare un problema di questo tipo come un problema di flusso minimo su un **grafo bipartito**:

Definizione 26.1: Grafo bipartito

Si dice **bipartito** un grafo dove i nodi N possono essere divisi in due insiemi disgiunti U e V, dove ogni arco collega un nodo U a un nodo V.

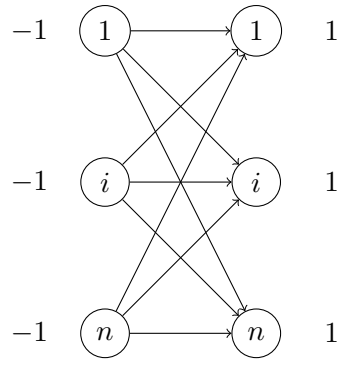
Graficamente, il grafo bipartito di un problema di trasporto ha la forma:



dove ogni nodo di **origine** O_i eroga o_i unità di flusso, e ogni nodo di domanda D_i richiede d_i unità di flusso.

26.1.2 Assegnamento di costo minimo

L'assegnamento di costo minimo può intedersi come un trasporto a **volume unitario**, cioè ogni nodo di origine eroga una singola unità di flusso, e ogni nodo di domanda richiede una singola unità di flusso. Chiaramente, per n nodi di domanda dovremo avere n nodi di origine (cioè, per ogni "spazio" assegnabile abbiamo bisogno di un possibile assegnamento). Questo si traduce sempre in un grafo bipartito, nella forma:



26.2 Cammini minimi

Abbiamo visto come l'ottimizzazione sui grafi può essere usata per risolvere problemi di flusso di costo minimo. Vediamo un'altro tipo di problema che può essere risolto attraverso una versione modificata del simplesso sui grafi: quello dei **cammini a costo minimo**.

In generale, un problema dei cammini a costo minimo è un problema di ottimizzazione sui grafi in forma:

$$\begin{cases} \min c^\intercal \cdot x \\ Ex = b \\ x \geq 0 \end{cases}$$

che dà i cammini minimi da un nodo r a tutti gli altri nodi $i \neq r$. Questo si ottiene scegliendo il vettore b appositamente:

$$b_i = \begin{cases} -(n-1), & i = r \\ 1, & i \neq r \end{cases}$$

cioè impostando il nodo r come una sorgente da $n-1$ unità, e ogni nodo di arrivo $i \neq r$ come un pozzo di 1 unità.

Notiamo che un albero di copertura T_r radicato in r rappresenta una base ammissibile del problema, in quanto porterà le $-(n-1)$ unità di flusso sugli $n-1$ nodi. Inoltre, il flusso di base asssociato a T_r sarà **non degenere**, cioè nessuno flusso sarà uguale a 0 . Quest'ultima proposizione potrebbe risultare poco intuitiva. Notiamo allora che:

- Ogni nodo non radice ha bilancio 1, ergo ogni arco deve portare almeno 1 unità di flusso su un nodo;
- Esiste solo un nodo sorgente, cioè la radice: questo significa che non esistono più strade per raggiungere lo stesso nodo (altrimenti una di esse si sarebbe potuta svuotare).

Possiamo quindi dire che il teorema di Bellman da solo basta a dimostrare l'ottimalità, e non occorrono regole anticiclo di Bland. Anzi, possiamo dire che conviene prendere l'arco con costo ridotto minimo, in quanto comporterà l'abbattimento maggiore del valore ottimo.

Infine, possiamo fare un'ottimizzazione significativa per quanto riguarda la rimozione degli archi in $\mathcal{C}^-$ (cioè quelli discordi all'arco introdotto (p,q)) nella fase di **rimozione dei cicli**. Si ha che a ogni passo dell'algoritmo la soluzione ammissibile considerata è un albero di copertura, ergo ogni nodo sarà raggiunto. L'algoritmo di Bellman restituirà quindi un modo *più efficiente* di quello previsto dall'albero di arrivare ad un dato nodo q, partendo da un nodo p. Avremo quindi che esiste sempre un nodo (i,q) **entrante** in q, cioè il modo di arrivare in q che era originariamente previsto dall'albero che abbiamo appena considerato. Inoltre, si avrà che questo arco sarà l'ultimo a portare a q, o almeno sarà a costo minore dei suoi precedenti (ogni precedente dovrà portare a q e a tutti gli eventuali nodi a cui arriva q). Ergo, (i,q) è a **flusso minimo**. Possiamo quindi semplificare la regola dell'arco uscente per la rimozione di cicli.

Formuliamo allora il **simplesso per cammini**:

Algoritmo 15 del simplesso per cammini

Input: un problema di cammini di costo minimo
Output: l'albero dei cammini minimi
Trova un albero T di radice r
ciclo:
Calcola il potenziale di base $\bar{\pi}^\intercal = c_T^\intercal E_T^{-1}$
Calcola i costi ridotti $c_{ij}^{\bar{\pi}} = c_{ij} + \bar{\pi}_{ij} - \bar{\pi}_{ij}$ per ogni arco
if $c_{ij}^{\bar{\pi}} \geq 0 \ \ \forall (i,j) \in L$ **then**
  Fermati, $\bar{x}$ è un albero dei cammini minimi
else
  Calcola l'arco entrante:

$$(p,q) = (i,j) \in L : c_{ij}^{\bar{\pi}} = \min\{c_{ij}^{\bar{\pi}}\}$$

  Chiama $\mathcal{C}$ il ciclo che l'arco (p,q) forma con gli archi in T
  Fissa un orientamento concorde a (p,q) su $\mathcal{C}$ e partiziona $\mathcal{C}$ in $\mathcal{C}^+$ archi concordi e $\mathcal{C}^-$ archi discordi a tale orientamento
end if
if $\mathcal{C}^- = \emptyset$ **then**
  Fermati, non esiste un albero dei cammini minimi
else
  Scegli come arco uscente l'unico arco $(i,q) \in T$ che entra in q
end if
Aggiorna l'albero come:

$$T = T \setminus \{(i,q)\} \cup \{(p,q)\}$$

Torna a ciclo

26.2.1 Cammini minimi multiobiettivo

Problema 26.1: Cammini minimi a due obiettivi

Vogliamo progettare un'applicazione di navigazione per dispositivi cellulari. L'applicazione usa una struttura dati a grafo per rappresentare località e le strade che collegano. Per ogni strada, si tiene conto di una distanza d_{ij} in KM, e di un costo c_{ij} al pedaggio. Si vuole trovare un algoritmo che trovi il percorso ottimo fra due località i e j.

Il problema presentato è un problema di ottimizzazione **multiobiettivo**: vogliamo *minimizzare* sia il costo che la distanza. Un'approccio è quello di impostare il problema di costo minimo sulle distanze, cioè come:

$$\begin{cases} \min d^\intercal \cdot x \\ Ex = b \\ x \geq 0 \end{cases}$$

e di introdurre un termine di *budget*, cioè un valore massimo C di costo che siamo disposti a pagare:

$$c^\intercal x \leq C$$

Questo approccio però non è propriamente ottimale, in quanto introducendo un nuovo vincolo, che potrebbe violare le condizoni di **interezza** rispettate dalla matrice di adiacenza, che sappiamo essere **unimodulare**. Per continuare a prendere archi interi, dovremmo obbligatoriamente introdurre un vincolo $x_{ij} \in \{0, 1\}$, ergo trasformare il problema in un problema di ILP. Esistono quindi metodi più efficienti, che però come per i multizaini non sono coperti dal corso.

27 Lezione del 12-11-24

27.1 Flusso di costo minimo capacitato

In realtà, il modello di flusso di costo minimo prevede, su ogni arco, un'altra quantità: la **portata** (o **capacità**) massima dell'arco. Avremo quindi che, oltre oltre agli n bilanci b_i e agli m costi c per n nodi e m archi, dovremo introdurre un nuovo vettore u di dimensione m per le **capacità superiori** di ogni arco.

Porremo quindi il nostro sistema come:

$$\begin{cases} \max c^\intercal x \\ Ex = b \\ 0 \leq x \leq u \end{cases}$$

dove l'ultimo vincolo è l'espressione in forma vettoriale di $0 \leq x_{ij} \leq u_{ij}\ \forall (i,j) \in A$.

Avevamo notato che un problema di flusso di costo minimo *non capacitato* si esprime agilmente in forma duale standard. Converrà quindi esprimere il vincolo delle capacità superiori come uguaglianza, per ricondursi allo stesso formato:

$$\begin{cases} \max c^\intercal x \\ Ex = b \\ x + w = u \\ x \geq 0 \\ w \geq 0 \end{cases}$$

Questa forma, con $w \in \mathbb{R}^m$, intende $x_{ij} + w_{ij} = u_{ij}$, cioè gli w_{ij} di ogni arco sono gli **scarti** dal valore di capacità massima. A w negativi si avranno $x > u$, ergo saremo oltre la portata massima, mentre a $w = 0$ avremo "fissato" la capacità al massimo per la x corrispondente.

Se volessimo restare nella forma matriciale, potremmo riscrivere il vincolo di capacità come:

$$Ix + Iw = u$$

sulla matrice identità I, e quindi esprimere a blocchi:

$$\begin{pmatrix} E & 0 \\ I & I \end{pmatrix} \begin{pmatrix} x \\ w \end{pmatrix} = \begin{pmatrix} b \\ u \end{pmatrix}$$

dove la prima matrice ha dimensioni $(m + n) \times 2n$. Denotiamo questa matrice M. Se avevamo dimostrato che il rango di $Ex = b$ è $n - 1$, cioè quello delle matrici di base (che danno alberi di copertura), sarà abbastanza intuitivo che il rango di questa nuova matrice sarà $m + n - 1$. In particolare, caratterizzare una base di questa matrice significherà trovare $m + n - 1$ colonne da selezionare per formare un minore invertibile.

Teorema 27.1: Caratterizzazione di base di matrici di incidenza capacitate

Presa la matrice M di vincoli data da un problema di flusso di costo minimo capacitato, espressa in forma:

$$M = \begin{pmatrix} E & 0 \\ I & I \end{pmatrix}$$

si suppone di avere una **tripartizione** T, L, U degli archi del grafo (cioè delle colonne della matrice), dove T è un *albero di copertura*. Si chiamano poi T', L', U' le colonne corrispondenti alle variabili di scarto w.
A questo punto, $B = T \cup U \cup T' \cup L'$ sarà un base.

Abbiamo che la tripartizione degli archi T, L, U è effettivamente una *esapartizione* delle colonne T, L, U, e T', L', U' con le w associate. Notiamo che M ha $2m$ colonne per definizione, e che queste sono distribuite ugualmente fra T, L, U e T', L', U', con il numero di colonne di T pari a quello di T', e via dicendo. Si ha quindi che scegliendo $U + T' + L'$ siamo a m colonne, a cui aggiungiamo le $n - 1$ di T (lo stesso ragionamento vale invertendo T e T'). Questo significa che la dimensione della base è $m + n - 1$, come volevamo.

Inoltre, visto che ci siamo limitati a moltiplicare per matrici identità, $\det(M) = det(E_T)$, ergo vale ancora il **teorema dell'interezza** e M è unimodulare (cioè il problema ha soluzioni intere).

27.1.1 Flussi di base capacitati

Vediamo quindi come calcolare il flusso di base corrispondente a una base:

$$(x(T, L, U), w(T', L', U')) = \left(E_T^{-1}(b - E_U u_U), 0, u_U, u_T - x_T, u_L, 0\right)$$

dove la notazione indica che x è composto dalle componenti in T, L, U, e w dalle componenti in T', L', U'.

La dimostrazione della formula è semplice: U' e L, essendo non in base, sono automaticamente zero. A questo punto, le U e L' corrispondenti sono determinate come i costi di flusso massimo, o i surplus massimi (conseguenti dal flusso nullo). Si calcolano infine le T rimaste applicando un'algoritmo greedy sulle foglie che tiene conto dei flussi determinati da U, e si calcolano le T' come gli scarti $u_T - x_T$ sulla capacità massima dati dal flusso appena determinato.

28 Lezione del 13-11-24

28.1 Potenziali di base capacitati

Abbiamo visto come calcolare i flussi di base, data un'opportuna tripartizione, sulla base T, U, T', L' di un problema di flusso minimo capacitato. Vediamo adesso come calcolare i potenziali di base. Si imposta innanzitutto il duale, cioè il **problema dei potenziali**:

$$\begin{cases} \max b^\intercal \pi + u^\intercal \mu \\ E^\intercal \pi + \mu \leq c \\ \mu \leq 0 \end{cases}$$

dove μ rappresenta gli **scarti** ai potenziali.

Notiamo che avremmo potuto usare la stessa matrice dei vincoli nel primale seguendo le stesse uguaglianze già viste sulle forme di matrici primali e duali:

$$\begin{cases} \min \begin{pmatrix} x & w \end{pmatrix}^\intercal \begin{pmatrix} c \\ 0 \end{pmatrix} \\ \begin{pmatrix} x & w \end{pmatrix}^\intercal \begin{pmatrix} E^\intercal & I \\ 0 & I \end{pmatrix} = \begin{pmatrix} b & u \end{pmatrix}^\intercal \\ \begin{pmatrix} x & w \end{pmatrix} \geq 0 \end{cases}$$

Lo stesso problema potrebbe essere stato espresso nella forma a blocchi, come avevamo visto sul primale:

$$\begin{cases} \max \begin{pmatrix} b^\intercal & u^\intercal \end{pmatrix} \begin{pmatrix} \pi \\ \mu \end{pmatrix} \\ \begin{pmatrix} E^\intercal & I \\ 0 & I \end{pmatrix} \begin{pmatrix} \pi \\ \mu \end{pmatrix} \leq \begin{pmatrix} c \\ 0 \end{pmatrix} \end{cases}$$

Vediamo quindi il calcolo vero e proprio. Si tratta il problema come qualsiasi altro problema in formato primale, cioè si rendono valide le equazioni date dalla base T, U, T', L'. Avremo quindi che $\mu_T = 0$ e $\mu_L = 0$. A questo punto troviamo i π dati da $E_T^\intercal \mu = c_T$, su T, L e U. Infine, abbiamo i μ su U, dati da $\pi_T E_U + \mu_U^\intercal = c^\intercal$. Cioè, riassumendo, secondo la stessa notazione usata per i flussi di base (si noti che il flusso in T è soluzione di un unico sistema):

$$(\pi(T), \mu(T'.L', U')) = \left(c_T^\intercal E_T^{-1}, 0, 0, c_U^\intercal - \pi^\intercal E_U \right)$$

Dai potenziali possiamo calcolare i costi ridotti, analogamente a come avevamo fatto sui flussi non capacitati:

$$c_{ij}^\pi = c_{ij} - \pi_i + \pi_j$$

e dimostrare una variante del teorema di Bellman:

Teorema 28.1: di Bellman capacitato

Supponiamo di avere una tripartizione T, L, U che generi un flusso di base ammissibile. Se la soluzione è anche ottima, vale riguardo ai costi ridotti:

$$\begin{cases} c_{ij}^\pi, \geq 0 & \forall (i,j) \in L \\ c_{ij}^\pi \leq 0, & \forall (i,j) \in U \end{cases}$$

29 Lezione del 14-11-24

29.1 Simplesso per flussi capacitati

Vediamo quindi come applicare l'algoritmo del simplesso ai problemi di flusso minimo capacitato. Si ha che, dal teorema di Bellman, gli archi entranti saranno quelli che violano i vincoli:

$$\begin{cases} c_{ij}^\pi \geq 0, & \forall (i,j) \in L \\ c_{ij}^\pi \leq 0, & \forall (i,j) \in U \end{cases}$$

Il fatto che un arco entrante può appartenere sia a L che a U significherà che dovremo fare delle considerazioni diverse in fase di elminazione dei cicli e in fase di inserzione dell'arco fra gli insiemi T, L, U

Troviamo quindi questo arco entrante $(p, q) \in L \cup U$, discriminando attraverso la regola anticiclo di Bland sull'ordine lessicografico degli archi. Vorremo eliminare il ciclo formato introducendovi $\vartheta \geq 0$ unità di flusso.

- Nel caso $(p, q) \in L$, introdurremo le unità ϑ nella direzione **concorde** al flusso per cercare di *riempire* l'arco.
- In caso contrario, con $(p, q) \in U$, introdurremo le unità ϑ nella direzione **discorde** al flusso, per cercare di *svuotare* l'arco (ricordiamo che gli archi in U sono quelli forzati pieni).

Scelta quindi una direzione per le unità ϑ, avremo che i flussi sul ciclo si evolvono come:

$$x_\vartheta = \begin{cases} \overline{x}_{ij} + \vartheta, & \forall (i,j) \in \mathcal{C}^+ \\ \overline{x}_{ij} - \vartheta, & \forall (i,j) \in \mathcal{C}^- \\ \overline{x}_{ij}, & \forall (i,j) \notin C \end{cases}$$

stabilite le partizioni concordi e discordi sul ciclo $\mathcal{C}^+$ e $\mathcal{C}^-$.

Notiamo che il flusso trovato aggiungendo o rimuovendo ϑ unità di flusso, che chiamiamo x_ϑ, rispetta Bellman, in quanto per un nodo posto in qualsiasi posizione fra archi concordi e discordi del ciclo, modificando i flussi secondo la regola riportata sopra non modificheremo mai il bilancio complessivo.

Questo si può dimostrare per enumerazione completa: preso un nodo n_c ad arbitrio su $\mathcal{C}$, si hanno i casi:

- n_c si trova fra due archi discordi: uno sarà discorde e l'altro concorde alla direzione di percorrenza, ergo avremo un unità di flusso in meno da un lato e una in più dall'altro.
- n_c si trova fra due archi concordi: questi cresceranno o diminuiranno di uno, ma in entrambi i casi l'unità in più (o in meno) di un arco dovrà essere fornita (ottenuta) dall'altro arco.

Ciò che cambierà invece saranno i costi, in quanto assunto che c^π_{pq} è uguale al costo del ciclo, si avrà che il percorso fatto sul ciclo senza (p, q) costerà la differenza di potenziale $c_{cic} = \pi_p - \pi_q$, e da $c_{pq} < c_{cic}$ sarà chiaramente più vantaggioso passare per (p, q).

Avremo quindi due condizioni di arresto per la variabile ϑ:

- Il primo caso è che un arco in $\mathcal{C}^-$ si annulli, cioè quello che avevamo visto sul flusso non capacitato, che si applica all'arco $(r, s) \in \mathcal{C}^-$ con $x_{rs} = \vartheta$
- Il secondo caso è che un arco in $\mathcal{C}^+$ arrivi a capienza totale, e si applica all'arco $(r, s) \in \mathcal{C}^-$ con $u_{rs} - x_{rs} = \vartheta$.

Notiamo che dai due casi non è possibile dare una definizione univoca di ϑ. Diciamo allora:

$$\vartheta^+ = \min\{u_{ij} - x_{ij},\ (i,j) \in \mathcal{C}^+\}$$

$$\vartheta^- = \min\{x_{\text{ij}},\ (i,j) \in \mathcal{C}^-\}$$

da cui:

$$\vartheta = \min\{\vartheta^+, \vartheta^-\}$$

cioè definiamo separatamente il ϑ^+ del primo arco saturo su $\mathcal{C}^+$, e il ϑ^- del primo arco vuoto su $\mathcal{C}^-$. Di conseguenza, ϑ sarà il minimo fra ϑ^+ e ϑ^-, cioè il primo che porta a una violazione dei vincoli.

Notiamo infine come ϑ può tendere a ∞: questo è il caso già visto dei cicli a costo negativo, e porta la soluzione del problema a $-\infty$.

Decideremo quindi di scegliere come arco uscente il primo arco (r, s), dalla regola anticiclo di Bland sull'ordine lessicografico degli archi, che ha $u_{rs} - x_{rs} = \vartheta$ per $(r, s) \in \mathcal{C}^+$ o $x_{rs} = \vartheta$ per $(r, s) \in \mathcal{C}^-$.

Resta quindi il problema di aggiornare le partizioni T, L, U sulla base degli archi (p, q) entrante e (r, s) uscente scelti. Dovremo effettivamente porci due domande:

- L'arco entrante (p, q) è vuoto ($\in L$) o saturo ($\in U$)?
- L'arco uscente (r, s) esce dopo essere svuotato ($\in \mathcal{C}^-$) o saturato ($\in \mathcal{C}^+$)?

Iniziamo a discriminare dalla prima domanda.

- $(p, q) \in L$:
  - $(r, s) \in \mathcal{C}^-$: l'arco entrante è vuoto e l'uscente esce svuotato. Vorremo spostare l'arco entrante in T e l'arco uscente in in L, cioè:

    $$T = T \setminus (r, s) \cup (p, q), \quad L = L \setminus (p, q) \cup (r, s)$$

  - $(r, s) \in \mathcal{C}^+$: l'arco entrante è vuoto e l'uscente esce saturato. Vorremo spostare l'arco entrante in T e l'arco uscente in in U, cioè:

    $$T = T \setminus (r, s) \cup (p, q), \quad L = L \setminus (p, q), \quad U = U \cup (r, s)$$

    Notiamo che in questa situazione può presentarsi il caso dove $(p, q) = (r, s)$, cioè l'arco (p, q) entra da vuoto e esce saturato, cioè semplicemente passa da L a U.

- $(p, q) \in U$:
  - $(r, s) \in \mathcal{C}^-$: l'arco entrante è pieno e l'uscente esce svuotato. Vorremo spostare l'arco entrante in T e l'arco uscente in in L, cioè:

    $$T = T \setminus (r, s) \cup (p, q), \quad L = L \cup (r, s), \quad U = U \setminus (p, q)$$

    Notiamo che in questa situazione può presentarsi il caso dove $(p, q) = (r, s)$, cioè l'arco (p, q) entra da pieno e esce svuotato, cioè semplicemente passa da U a l.
  - $(r, s) \in \mathcal{C}^+$: l'arco entrante è pieno e l'uscente esce saturato. Vorremo spostare l'arco entrante in T e l'arco uscente in in U, cioè:

    $$T = T \setminus (r, s) \cup (p, q), \quad U = U \setminus (p, q) \cup (r, s)$$

Possiamo quindi formulare l'algoritmo:

Algoritmo 16 del simplesso per flussi capacitati

Input: un problema di flusso di costo minimo capacitato
Output: la soluzione ottima
Trova una partizione degli archi (T, L, U) con T albero di copertura che genera un flusso ammissibile
ciclo:
Calcola il flusso di base capacitato:

$$\bar{x} = \left(E_T^{-1}(b - E_U u_U), 0, u_U, u_T - x_T, u_L, 0\right)$$

e il potenziale di base capacitato:

$$\bar{\pi} = \left(c_T^\intercal E_T^{-1}, 0, 0, c_U^\intercal - \pi^\intercal E_U\right)$$

Calcola i costi ridotti $c_{ij}^{\bar{\pi}} = c_{ij} + \bar{\pi}_{ij} - \bar{\pi}_{ij}$ per ogni arco
if Bellman è soddisfatto, cioè:

$$c_{ij}^{\overline{\pi}} \geq 0 \ \ \forall (i,j) \in L$$

$$c_{ij}^{\overline{\pi}} \leq 0 \ \ \forall (i,j) \in U$$

then
Fermati, $\bar{x}$ è un flusso ottimo e $\bar{\pi}$ è un potenziale ottimo
else
Calcola l'arco entrante:

$$(p,q) = \min\left\{\{(i,j) \in L : c_{ij}^{\bar{\pi}} < 0\} \cup \{(i,j) \in U : c_{ij}^{\bar{\pi}} > 0\}\right\}$$

stabilito l'ordinamento lessicografico degli archi
Chiama $\mathcal{C}$ il ciclo che l'arco (p,q) forma con gli archi in T
Fissa un orientamento concorde a (p,q) su $\mathcal{C}$ se $(p,q) \in L$, e discorde se $(p,q) \in U$ e partiziona $\mathcal{C}$ in $\mathcal{C}^+$ archi concordi e $\mathcal{C}^-$ archi discordi a tale orientamento
end if
Calcola:

$$\vartheta^+ = \min\{u_{ij} - \bar{x}_{ij} : (i,j) \in \mathcal{C}^+\}$$

$$\vartheta^- = \min\{\bar{x}_{ij} : (i,j) \in \mathcal{C}^-\}$$

$$\vartheta = \min\{\vartheta^+, \vartheta^-\}$$

Notiamo che le stesse ottimizzazioni che avevamo visto sul simplesso per i flussi valgono per il simplesso per i flussi capacitati.

if $\vartheta = \infty$ **then**
Fermati, il flusso di costo minimo ha valore $-\infty$
else
Trova l'arco uscente:

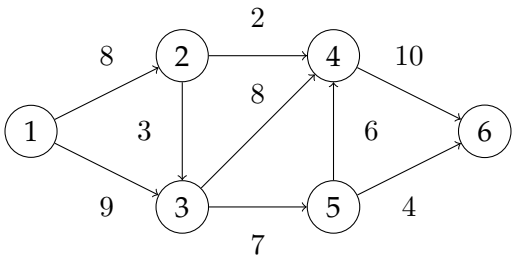
$$(r,s) = \min\left\{\{(i,j) \in \mathcal{C}^+ : u_{ij} - \bar{x}_{ij} = \vartheta\} \cup \{(i,j) \in \mathcal{C}^- : \bar{x}_{ij} = \vartheta\}\right\}$$

stabilito un ordinamento lessicografico degli archi
end if
Aggiorna le partizioni come:
if $(p,q) \in L$ **then**
if $(r,s) \in \mathcal{C}^-$ **then**
$T = T \setminus (r,s) \cup (p,q)$, $L = L \setminus (p,q) \cup (r,s)$
else
if $(p,q) = (r,s)$ **then**
$L = L \setminus (p,q)$, $U = U \cup (p,q)$
else
$T = T \setminus (r,s) \cup (p,q)$, $L = L \setminus (p,q)$, $U = U \cup (r,s)$
end if
end if
else
if $(r,s) \in \mathcal{C}^-$ **then**
if $(p,q) = (r,s)$ **then**
$L = L \cup (p,q)$, $U = U \setminus (p,q)$
else
$T = T \setminus (r,s) \cup (p,q)$, $L = L \cup (r,s)$, $U = U \setminus (p,q)$
end if
else
$T = T \setminus (r,s) \cup (p,q)$, $U = U \setminus (p,q) \cup (r,s)$
end if
end if
Torna a ciclo

30 Lezione del 18-11-24

30.1 Problema di flusso massimo

Poniamo di avere un grafo su cui riportiamo solamente le capacità superiori u_{ij} sui singoli archi:



Prendiamo due nodi s e t, e cerchiamo di **massimizzare** il flusso da s a t. L'idea fondamentale che il flusso che parte da s dovrà essere uguale al flusso che arriva in t. Potremo partire da un flusso ammissibile qualsiasi, cioè che si limita a rispettare le capacità:

$$x = (2, 4, 0, 2, 0, 4, 2, 0, 4)$$

Notando che le capacità sugli *ultimi* archi influenzano quelle sugli archi precedenti (ad esempio, $(3, 5)$ è limitato a 4 da $(5, 6)$ con $u_{56} = 4$). Poniamo allora, più intelligentemente, un problema di PL:

$$\begin{cases} \max v \\ Ex = b \\ 0 \leq x \leq u \end{cases}$$

dove E è la matrice di incidenza della rete, e i bilanci b_i stessi dipendono dalla variabile v:

$$b_i = \begin{cases} -v, & i = s \\ 0, & i \neq s \wedge i \neq t \\ v, & i = t \end{cases}$$

Potremmo avere dubbi sul fatto che la variabile v compare nei bilanci b. Scriviamo per esteso le uguaglianze dei vincoli:

$$\begin{cases} -x_{12} - x_{13} = -v \\ x_{12} - x_{23} - x_{24} = 0 \\ x_{13} + x_{23} - x_{34} - x_{35} = 0 \\ x_{24} + x_{34} + x_{54} - x_{46} = 0 \\ x_{35} - x_{54} - x_{56} = 0 \\ x_{46} + x_{56} = v \end{cases}$$

Notiamo che questo problema è effettivamente un problema di flusso minimo (massimo) su $n+1$ variabili per n nodi, dove la n-esima variabile è proprio il flusso v. Inoltre, con capacità massime intere, anche v sarà necessariamente intero (dato da somma di interi) e quindi la matrice **unimodulare**, con la conseguenza già vista che PL $=$ ILP.

Inoltre, possiamo portare i termini v di b a sinistra delle rispettive equazioni, ottenendo effettivamente la matrice:

$$\begin{pmatrix} & 1 & \\ E & 0 & \\ & -1 & \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} = 0$$

cioè dove si è introdotto la un **arco fittizio**, quello che parte da t e arriva in s. Capovolgendo la funzione obiettivo (e riportando in vista i termini di costo nullo su ogni arco x_{ij}), otteniamo quindi:

$$\begin{cases} \min 0 \cdot x - v \\ \begin{pmatrix} & 1 & \\ E & 0 & \\ & -1 & \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} = 0 \\ 0 \leq x \leq u \end{cases}$$

dove per l'ultimo arco fittizio la capacità massima u è un M molto grande (o ∞).

Questo è un problema di flusso di costo minimo capacitato che include l'arco fittizio come unico arco a costo diverso da 0 (per giunta negativo), cioè che è "costretto" a imporre flusso massimo da t ad s, e visto che tutti i nodi sono a bilancio 0, a riportarlo in direzione opposta da s a t lungo i nodi della rete vera e propria.

A questo punto possiamo proporre la soluzione ammissibile:

$$(x, v) = (2, 9, 0, 2, 5, 4, 7, 0, 4, 11)$$

da cui si ricavano le partizioni, controllando quali archi si svuotano e quali saturano:

$$T = \{(1,2),(3,4),(3,5),(4,6),(6,1)\}, \quad L = \{(2,3),(5,4)\}, \quad U = \{(1,3),(2,4),(5,6)\}$$

Possiamo quindi ricavare il potenziale dell'albero di copertura, notando che dividerà necessariamente i nodi in partizione di potenziale 0 e potenziale 1:

$$\pi = (0, 0, 1, 1, 1)$$

e calcolare i costi ridotti, che troviamo negativi su $(2,3) \in L$, quindi arco entrante:

$$c^{\pi}_{23} = 0 + 0 - 1 = -1$$

che è quanto ci aspettavamo, in quanto chiaramente $v = 14$ (dagli archi entranti nel nodo 6).

Procediamo quindi con l'eliminazione del ciclo, distinguendo innanzitutto le pratizioni $\mathcal{C}^+$ e $\mathcal{C}^-$:

$$\mathcal{C}^+ = \{(1,2),(2,3),(3,4),(4,6),(6,1)\}, \quad \mathcal{C}^- = \emptyset$$

Su $\mathcal{C}^-$ mettiamo $\vartheta^- = \infty$ (essenzialmente vogliamo considerare solo ϑ^+). Calcoliamo allora ϑ^+ dagli $u_{ij} - x_{ij}$, tenendo conto che la capacità u di $(6,1)$ (arco fittizio) è ∞:

$$\vartheta^+ = \max\{6, 3, 3, 3, \infty\}$$

da cui prendiamo $\vartheta^+ = 3$ e l'arco uscente $(2,3)$ per l'ordinamento lessicografico. Notiamo di essere nel caso particolare dove l'arco *entra* ed *esce* (in questo caso si sposta da L a U).

Aggiungendo quindi il ϑ agli archi in $\mathcal{C}^+$ otteniamo il flusso:

$$(x, v) = (5, 9, 3, 2, 8, 4, 10, 0, 4, 14)$$

da cui $v = 14$, come ci aspettavamo. Vedremo in seguito un algoritmo per il calcolo del flusso massimo su reti più efficiente dell'applicazione diretta del simplesso.

31 Lezione del 19-11-24

31.0.1 Algoritmo di Ford-Fulkerson

L'algoritmo di **Ford-Fulkerson** (precisamente nella variante di *Edmonds-Karp*) è un algoritmo per il calcolo del flusso massimo su reti.

Iniziamo con alcune definizioni, prima fra tutte quella di **taglio di rete**:

Definizione 31.1: Taglio di rete

Dato un problema di flusso massimo, un taglio di rete è una partizione dei nodi in due sottoinsiemi N_s e N_t, quindi con $N_s \cup N_t = N$ e $N_s \cap N_t = \emptyset$, con il vincolo aggiunto che la sorgente di flusso s deve trovarsi $\in N_s$, e la destinazione del flusso t deve trovarsi $\in N_t$.

Possiamo quindi dare la definizione di **arco diretto del taglio**:

Definizione 31.2: Arco diretto del taglio

Dato un taglio, si dicono archi diretti del taglio tutti gli archi (i, j) che hanno $i \in N_s$ e $j \in N_t$.

con enfasi sul fatto che un arco diretto del taglio va da N_s a N_t, e **non** viceversa.

Infine, su questi archi diretti del taglio definiamo la **portata**:

Definizione 31.3: Portata del taglio

Dato un taglio (N_s, N_t), la portata $\mu(N_s, N_t)$ viene definita come:

$$\mu(N_s, N_t) = \sum_{(i,j) \in A^\intercal} u_{ij}$$

Notiamo come ogni taglio di rete rappresenta un limite superiore sul flusso massimo:

$$\overline{x} \leq \mu(N_s, N_t)$$

Possiamo quindi dimostrare il teorema:

Teorema 31.1: Massimo flussimo - minimo taglio

Il flusso massimo di un problema di flusso massimo con soluzione $\neq -\infty$ è uguale al taglio di portata minima:

$$\overline{v} = \overline{u}(N_s, N_t) = \min_{s,t} \mu(N_s, N_t)$$

Notiamo come le ultime due equazioni ricalcano rispettivamente la **dualità debole** e la **dualità forte**: si può effettivamente dimostrare che il problema del taglio minimo è il **duale** del problema di flusso massimo (effettivamente vogliamo trovare un *limite superiore* del flusso, che è lo stesso procedimento che avevamo adottato, in quel caso adattando le diseguaglianze, nel ricavare il duale nei problemi di LP).

Chiaramente è improbabile calcolare il taglio minimo per enumerazione completa: togliendo il fatto che le partizioni dei tagli lavorano su $n - 2$ anziché n nodi (in quanto

ci sono i vincoli $s \in N_s$ e $t \in N_t$), il numero di partizioni va comunque come $\sim 2^n$, che è quindi a complessità **esponenziale**.

Vediamo allora quale algoritmo possiamo usare per ricavare i tagli minimi (e quindi i flussi massimi) in maniera efficiente.

31.0.2 Definizione dell'algoritmo

Prendiamo in considerazione un grafo e il suo **grafo residuo**, cioè il grafo che complementa tutti gli archi (diretti) con un arco (diretto) identico ma di verso opposto. Chiameremo *archi reali* gli archi che fanno effettivamente parte del grafo, e *archi fittizi* gli archi che introduciamo sul grafo residuo.

Per ogni arco (i, j), reale o fittizio, definiamo la **capacità residua** come:

- Archi reali: $r_{ij} = u_{ij} - \overline{x}_{ij}$
- Archi fittizi: $r_{ji} = \overline{x}_{ij}$

assunto nelle definizioni che (i, j) sia l'arco reale e di conseguenza (j, i) l'arco fittizio.

A parole, la capacità residua dell'arco *reale* è *quanto posso ancora spedire sull'arco*, mentre la capacità residua dell'arco *fittizio* è *quanto ho già spedito sull'arco vero*, da cui:

$$r_{ij} + r_{ji} = u_{ij} - \overline{x}_{ij} + \overline{x}_{ij} = u_{ij}$$

cioè la somma delle capacità residue su entrambi gli archi nelle due direzioni del grafo residuo dà sempre la capacità massima sull'arco reale nella stessa posizione del grafo originale.

Definiamo allora il concetto di **cammino aumentante** sul grafo:

Definizione 31.4: Cammino aumentante

Dato un problema di flusso massimo, chiamiamo cammino aumentante C_{aum} sul grafo residuo un qualsiasi cammino orientato da s a t formato da archi con capacità residue $r_{ij} > 0$.

Di un dato cammino aumentante C_{aum}, il dato interessante è la **portata** di C_{aum}:

$$\delta = \min_{i,j \in C_{aum}} \{r_{ij}\}$$

Il primo passo dell'algoritmo di Ford-Fulkerson sarà allora selezionare un cammino aumentante e calcolarne la portata δ. In seguito vorremo spedire queste δ unità di flusso lungo il cammino, quindi applicando la regola:

$$\overline{x}' = \begin{cases} \overline{x}_{ij} + \delta, & \forall (i, j) \in C_{aum} \\ \overline{x}_{ij}, & \forall (i, j) \notin C_{aum} \end{cases}$$

A questo punto basta ricalcolare le varie capacità residue r_{ij} e ripetere il passaggio. Chiaramente, quando non si riuscirà più a trovare cammini aumentanti, cioè archi con capacità residue > 0, significherà che avremo trovato la soluzione ottima.

Algoritmo 17 di Ford-Fulkerson

Input: un problema di flusso massimo
Output: la soluzione ottima
ciclo:
Stabilisci un cammino aumentante C_{aum} e calcola la portata δ.
if $C_{aum} = \emptyset$ **then**
Fermati, sei all'ottimo
end if
Aggiorna il flusso $\overline{x}$ secondo la regola:

$$\overline{x}' = \begin{cases} \overline{x}_{ij} + \delta, & \forall (i,j) \in C_{aum} \\ \overline{x}_{ij}, & \forall (i,j) \notin C_{aum} \end{cases}$$

Ricalcola le capacità residue r_{ij}
Torna a ciclo

Si presenta un algoritmo per il calcolo dei cammini aumentanti, detto **algoritmo della croce**:

Algoritmo 18 della croce

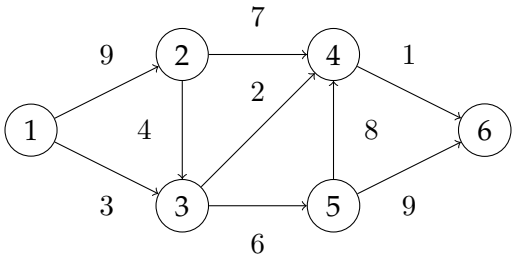
Input: il grafo residuo di un problema di flusso massimo
Output: un cammino aumentante C_{aum}
Inizializza due vettori, `closed` e `open`
Inserisci s, il nodo di partenza, in `closed`
Inserisci tutti i nodi raggiungibili da `closed`, cioè quelli che sono collegati con un arco diretto dal nodo 1 con costo residuo > 0, in `open`
ciclo:
Prendi il primo elemento di `open` e mettilo in `closed`
Inserisci tutti i nodi raggiungibili da `closed`, cioè quelli che sono collegati con un arco diretto dal nodo in `closed` con costo residuo > 0
if non hai raggiunto t; cioè il nodo di arrivo **then**
Torna a ciclo
end if
Ripercorrendo i nodi in `closed`, partendo da quello di arrivo, hai il cammino aumentate C_{aum}

Notiamo come, nell'applicazione dell'algoritmo della croce, è necessaria la presenza degli archi residui in entrambe le direzioni: potrebbe infatti succedere che un arco abbia flusso aggiuntivo δ nella direzione opposta a quella canonica. Questo significherà che il flusso che spingiamo nella direzione opposta andrà a sottrarsi al flusso già presente sull'arco (in altre parole, l'algoritmo di Ford-Fulkerson può "cambiare idea" nell'esplorazione dello spazio delle soluzioni).

32 Lezione del 20-11-24

32.1 Algoritmo di Dijkstra

Torniamo sul problema dei **cammini di costo minimo**. Prendiamo il grafo:



Un (celebre) algoritmo per il problema è quello di **Dijkstra**.

Algoritmo 19 di Dijkstra

Input: un problema di cammini minimi con costi $c_{ij} > 0$
Output: l'albero dei cammini minimi
Etichetta ogni nodo come $\pi = (0, +\infty, ..., +\infty)$, assunto il primo nodo come partenza
Associa un predecessore ad ogni nodo come $P = (1, 0, ..., 0)$
Poni $U = N$
while $U \neq \emptyset$ **do**
  Prendi il nodo $n \in U$ di etichetta π minima, rimuovilo da U, e prendi la stella uscente $\mathrm{FS}(n)$ di n
  for $\forall j \in \mathrm{FS}(n)$ **do**
    if $\pi_j \geq \pi_i + c_{ij}$ **then**
      Poni il predecessore $P_j = i$
      Poni l'etichetta $\pi_j = \pi_i + c_{ij}$
    end if
  end for
end while

Le etichette π rappresenteranno la **distanza minima** dal nodo di partenza a ogni nodo. Ad algoritmo compiuto (quando N è vuoto) la catena dei predecessori da ogni nodo a quello di partenza forma i cammini minimi.

Sull'esempio riportato sopra, il primo passaggio è:

$$i = 1, \quad N = \{2, 3, 4, 5, 6\}$$

Da cui si calcola quindi la stella uscente $FS(1) = \{2, 3\}$ Prendiamo entrambi gli indici:

- $j = 2$: $\pi_2 \geq \pi_1 + c_{12} \rightarrow \begin{cases} \pi_2 = 8 \\ p_2 = 1 \end{cases}$
- $j = 3$: $\pi_3 \geq \pi_1 + c_{13} \rightarrow \begin{cases} \pi_3 = 3 \\ p_2 = 1 \end{cases}$

Da cui, all'inizio del secondo passaggio, etichette e predecessori sono:

$$\pi = (0, 8, 3, +\infty, +\infty, +\infty)$$

$$p = (1, 1, 1, 0, 0, 0)$$

L'algoritmo, con complessità linearitmica $O(n \log n + m \log n)$ su n nodi a ramificazione m, risulta più efficiente della risoluzione diretta dei problemi dei cammini minimi come problemi di flusso.

33 Lezione del 26-11-24

33.1 Introduzione alla programmazione non lineare

Problema 33.1: di programmazione non lineare

In una nuova linea di produzione sono integrati 6 robot, ciascuno dei quali può rotare di 360 gradi attorno all'asse verticale. Le aree di lavoro di ciascun robot non devono sovrapporsi, in modo da evitare collisioni fra di essi. Inoltre, i robot devono essere collegati fra di loro attraverso cavi in fibra ottica, e poichè questi sono costosi, è necessario minimizzare la distanza da coprire.

I raggi di lavoro dei robot sono, poniamo, i seguenti:

Robot	Raggio
1	120
2	80
3	100
4	70
5	45
6	120

Il problema sarà quindi quello di individuare i 6 punti sul piano $(x_1, y_1), (x_2, y_2)$, ..., (x_6, y_6) che corrispondono alle posizioni degli assi verticali di ogni robot, imponendo il vincolo di distanza fra due robot adiacenti i e j di $r_i + r_j$. La funzione obiettivo sarà quindi la quantità di cavo in fibra ottica, cioè la distanza fra ogni coppia i, j di robot. Possiamo quindi disporre un modello:

$$\begin{cases} \max \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} + \sqrt{(x_2 - x_3)^2 + (y_2 - y_3)^2} + ... + \sqrt{(x_5 - x_6)^2 + (y_5 - y_6)^2} \\ d_{ij} \geq r_j + r_j, \quad \forall i = 1, 2, ..., 6, \quad \forall j = 1, 2, ..., 6, \quad i < j \end{cases}$$

su 12 variabili e $\frac{n(n-1)}{2}\big|_{n=6} = 15$ vincoli. Nelle prossime lezioni vedremo come risolvere questo tipo di problemi, notando per ora che tutte le funzioni che abbiamo generato finora sono **quadratiche** tolte le radici (che possiamo fare conservando massimi e minimi).

In generale, nella programmazione non lineare, vorremo avere le nostre funzioni $\in C^1$ in modo da poter derivare, cioè ricavare il gradiente e stabilire punti di massimo e di minimo, attraverso il teorema di Fermat:

Teorema 33.1: di Fermat

Sia data una funzione $f : \mathbb{R}^n \to \mathbb{R}$, con adeguate condizioni di continuità. Si ha che in ogni punto di massimo o minimo locale $\bar{x}$ vale:

$$\nabla f(\bar{x}) = 0$$

Addirittura, fosse la funzione $\in C^2$ potremmo ricavare l'Hessiana come ulteriore discriminante, ricordando che *Hessiana definita positiva* significa **punto di minimo**, e viceversa *Hessiana definita negativa* significa **punto di massimo** (con le opportune considerazioni di definizione positiva/negativa stretta e non).

Vediamo quindi di fare una classificazione di funzioni utili ai fini dell'ottimizzazione.

33.1.1 Funzioni quadratiche

Diamo quidi una forma generale di polinomi di secondo grado su cui fare ottimizzazione. Una funzione tipo potrebbe essere:

$$f(x) = \frac{1}{2}x^\intercal Qx + c^\intercal x$$

imponendo, vedremo poi come mai, la matrice Q definita positiva.

Di cui potrebbe essere un esempio:

$$f(x) = ax_1^2 + bx_1x_2 + cx_2^2 + dx_1 + ex_2$$

Vediamo come trovare la matrice Q e il vettore c. Nel caso in due variabili, che continuiamo per la convenzione della ricerca operativa a chiamare x_1 e x_2, avremo per calcolo diretto, spezzando le componenti in Q e in C:

- Q:

$$\frac{1}{2}x^\intercal Qx = \frac{1}{2}\begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{pmatrix}\begin{pmatrix} x_1 & x_2 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} q_{11}x_1 & q_{12}x_2 \\ q_{21}x_1 & q_{22}x_2 \end{pmatrix}$$

$$= \frac{1}{2}\left(q_{11}x_1^2 + (q_{12} + q_{21})x_1x_2 + q_{22}x_2^2\right)$$

  sarà quindi:

$$\begin{cases} q_{11} = 2a \\ q_{12} = b \\ q_{21} = b \\ q_{22} = 2c \end{cases}$$

- c: valgono le stesse regole viste sulle lineari, cioè nel nostro esempio:

$$\begin{cases} c_1 = d \\ c_2 = e \end{cases}$$

Abbiamo quindi stabilito che esiste una corrispondenza fra polinomi di secondo grado e matrici quadrate simmetriche di dimensione 2, dove la definizione positiva indica la presenza di un solo minimo globale della quadratica.

Si può poi dimostrare che in generale esiste una corrispondenza fra **funzioni polinomiali di secondo grado** e **matrici quadrate simmetriche** di qualsiasi dimensione.

Notiamo che, stabilita la Q e la c, il calcolo di gradiente risulta immediato in quanto:

$$\begin{cases} \nabla f = Qx + c \\ Hf = Q \end{cases}$$

da cui si spiega l'$\frac{1}{2}$ accanto a Q, che "normalizza" il modulo dell'Hessiana.

33.1.2 Funzioni convesse

Diamo innanzitutto una definizione:

Definizione 33.1: Funzione convessa

Una funzione convessa è qualsiasi funzione che rispetta:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$$

con $\forall \lambda \in [0,1], \forall x, \forall y \in \mathbb{R}^n$.

la definizione di funzione convessa stabilisce essenzialmente per una qualsiasi retta intercettante la funzione f, in qualsiasi punto compreso fra le intercette la funzione raggiunge un valore minore o uguale al valore della retta calcolato nello stesso punto.

Una proprietà importante delle funzioni convesse è che il gradiente è dato da una funzione **monotona crescente**, cioè $f(x) < f(y)\ \forall x < y$, e quindi che l'Hessiana è **semidefinita positiva** (≥ 0 nel caso scalare).

Una conseguenza immediata di quest'ultima considerazione è che in una funzione convessa, ogni punto dove $\nabla f(x) = 0$ è necessariamente **minimo**.

Notiamo alcune altre proprietà delle funzioni convesse:

- Una funzione convessa può non avere minimo: si pensi all'esponenziale, che tende a 0 senza raggiungerlo mai, e quindi ha solo un *limite inferiore*;
- Il minimo locale di una funzione convessa è anche minimo globale;
- Le funzioni convesse non hanno selle.

33.1.3 Funzioni coercive

Definiamo infine un'ultima classe, data da:

Definizione 33.2: Funzione coerciva

Una funzione coerciva è una funzione che rispetta il limite:

$$\lim_{|x| \to +\infty} f(x) = +\infty$$

Possiamo quindi essere sicuri che una funzione coerciva non ha massimo globale, ma al massimo solo massimi locali.