Ricerca Operativa 2024/2025

Davide Squeri

19/02/2025

Preambolo

Questi appunti sono stati presi durante l'anno accademico 2024/2025, sotto l'insegnamento del professor Pappalardo, dal quale non sono stati revisionati e pertanto non costituiscono materiale ufficiale.

Per qualsiasi domanda potete contattarmi su instagram (@davidesqueri), su github (@learningDane), via mail (squeridavide@gmail.com) e ovviamente via whatsapp (non metterò però qua il mio numero).

Consiglio la visualizzazione di questi appunti su github pages (`learningDane.github.io/uninotes`), dove potete trovare la versione interattiva con i collegamenti simil-wiki. Ci sono tutti i miei appunti universitari, quelli di Ricerca Operativa compresi.

Consiglio inoltre VIVAMENTE l'utilizzo del mazzo di Flash Cards Anki che ho caricato sul github, vi ho raccolto ogni possibile domanda da orale e mi è servito tremendamente nella mia preparazione dell'esame. Potete trovarlo nella cartella di ricerca operativa o al seguente link: `https://github.com/Guray00/IngegneriaInformatica/blob/master/SECONDO%20ANNO/I%20SEMESTRE/Ricerca%20Operativa/ANKI%20Flash%20Cards%20Ric%20Operativa%20Squeri.xml`

Scaricate il file e importatelo in AnkiAPP, non ho provato su altri client e pertanto non assicuro niente.

Informazioni Generali

L'esame si compone di scritto e orale, al quale si è ammessi con un minimo di 15 allo scritto. Scritto e Orale sono divisi in 4 problemi, uno per ogni parte del corso.

Il libro consigliato è "Ricerca Operativa" di Massimo Pappalardo e Mauro Passacantando, edito da Unipi. Ogni parte del corso dura circa 3 settimane, per 8 ore a settimana equivale a 90 ore totali.

Prerequisiti:

1. parte 1,2,3 = Algebra Lineare: operazioni su matrici, risoluzione sistemi lineari, inversa di matrice, teorema del rango/numero di soluzione di sistemi lineare.
2. parte 4 = Analisi Matematica II

Useremo MatLab, in particolare la optimization toolbox.

Allo scritto possiamo usare il nostro calcolatore, con matlab e appunti, basta non comunicare con altri o utilizzare internet.

Argomenti

1 Introduzione

La *Ricerca Operativa* è una ottimizzazione in forma deterministica.

È una branca della matematica applicata in cui problemi decisionali complessi vengono analizzati e risolti mediante modelli matematici e metodi quantitativi avanzati (ottimizzazione, simulazione, ecc.) come supporto alle decisioni stesse.

Gli argomenti principali della Ricerca Operativa includono:

- Modelli matematici di ottimizzazione per processi decisionali.
- Teoria e metodi di Programmazione Matematica: Lineare (PL), Lineare Intera (PLI), Lineare su Reti e Non Lineare (PNL).
- Toolbox Optimization di MATLAB per la risoluzione di problemi di ottimizzazione.
- Problemi specifici come produzione, assegnamento, trasporto, caricamento, localizzazione, "bin packing", commesso viaggiatore, flusso di costo minimo su reti, cammini minimi e flusso massimo.

2 Programmazione Lineare (PL)

Un problema di programmazione lineare segue il seguente Modello Matematico:

$$\begin{cases} \min / \max \ c^T \cdot x \\ A \cdot x \leq b \\ B \cdot x \geq d \\ C \cdot x = e \\ x \in Z^n \end{cases}$$

Un problema di Programmazione Lineare consiste nel trovare il massimo o il minimo di una funzione lineare soggetta ad un insieme finito di vincoli lineari di disuguaglianza o di uguaglianza.

2.1 Forme Standard

Una forma standard è un formato in cui può essere portato ogni problema attraverso **trasformazioni equivalenti**. Ciò si fa per semplicizzare oppure perché programmi come **MatLab** accettano solo alcuni formati standard.

2.1.1 Formato Primale Standard

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

2.1.2 Formato Duale Standard

$$\begin{cases} min \ b^T y \\ A^T y = c \quad \text{oppure} \quad y^T A = c^T \\ y \geq 0 \end{cases}$$

2.1.3 Formato Linprog Standard

$$\begin{cases} min \ c^T x \\ Ax \leq b \\ A_{eq} = b_{eq} \\ LB \leq x \leq UB \end{cases}$$

2.2 Trasformazioni Equivalenti

- i $\geq$ diventano $\leq$ (inverto di segno)
- i = diventano un sistema $\begin{cases} \leq \\ \leq \end{cases}$, ovvero raddoppio i vincoli
- i min diventano max, poiché $\overline{x} \in argmax \ f \leftrightarrow \overline{x} \in argmax \ -f$

- $Ax \leq b$ diventano un sistema $\begin{cases} Ax + S = b \\ S \geq 0 \end{cases}$ con S dicesi *slack*, ovvero AGGIUNGO VARIABILE

2.3 Soluzione Ottima

La Regione Ammissibile di un problema (P) di programmazione lineare è un Poliedro P.

1. Il max se esiste, appartiene al bordo (Teorema di Fermat)
2. Il max può essere $+\infty$ (con P illimitati)
3. La Soluzione Ottima non è per forza unica
4. P può essere vuoto ($\rightarrow$ ranking dei vincoli)
5. Almeno un vertice è soluzione ottima
6. se la regione ammissibile si restringe i minimi salgono (Rilassamento)

Cono di Competenza = insieme di vettori c che hanno tra le soluzioni ottimali lo stesso vertice.

Si dice cono di competenza relativo ad un vertice. Proposizione:
$(P) \in PL$:

1. se max $cx = -\infty \implies$ ha P vuoto
2. può avere P illimitato
3. se ha 2 soluzioni ottime $\implies$ ha illimitate soluzioni ottime

2.4 Teorema Fondamentale dei Problemi PL

Se P è limitato e non vuoto allora (almeno) un vertice di P è ottimo.

Sia dato (P) $\begin{cases} \max c^T \cdot x \\ Ax \leq b \end{cases}$ e la sua Rappresentazione di Weyl.

Supponiamo che il Poliedro P sia non vuoto e che la soluzione sia limitata:
Allora

$$\exists r \in \{1, .., k\} : max_{x \in P}\ c^T x = c^T v^r$$

Osservazioni

1. se P è limitato allora $E = \{\vec{0}\}$
2. se P ha vertici $V = \{vert\ P\}$:

tutti i poliedri limitati hanno vertici
Tutti i poliedri senza vertici sono: semipiani, strisce e rette.
Quindi tolti questi 3 casi patologici:

1. poliedro vuoto
2. poliedro senza vertici
3. poliedro illimitato

Allora almeno un vertice è soluzione ottima.

2.5 Linprog

Per trovare una soluzione ottima di un Problema di Programmazione Lineare, MatLab offre linprog. Risolve i problemi nella forma linprog standard:

$$\begin{cases} min\ c^T x \\ Ax \leq b \\ A_{eq} = b_{eq} \\ LB \leq x \leq UB \end{cases}$$

dove LB e UB sono i vettori dei lower e upper bound.

2.5.1 Esempio

$$\begin{cases} max\ x_1 + x_2 \\ 3x_1 + 5x_2 \geq 7 \\ 2x_1 + 6x_2 - 9x_3 = 9 \\ x_1, x_2, x_3 \geq 0 \\ x_2 \leq 8 \end{cases}$$

Digitare: (ordine e nomi alle matrici non contano, conta solo ordine nel comando linprog(...))

```
Matlab
>> c=[-1 0 -1]
>> ub=[ ; 8 ; ]
>> lb=[0;0;0]
>> beq=[9]
>> b=[-7 ; 3]
>> a=[-3 -5 0 ; 1 0 1]
>> aeq=[2 6 -8]
>> [x,v]=linprog(c,a,b,aeq,beq,lb,ub)
>> appare soluzione: x=(val ottimo)
```

2.6 Teoria della Dualità

Dato un Problema di Programmazione Lineare (PL) in Problema di Programmazione Lineare in Formato Primale Standard ne individuiamo il suo **complementare duale**:

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

Questa è una operazione di dualità, trovare il duale di un primale; Inoltre il duale di un duale è il suo primale!

2.6.1 Teorema della Dualità Forte/degli Scarti complementari

Sia (P) un problema PL con Duale (D), se x, y sono soluzioni ammissibili di $(P), (D)$ e inoltre:

1. ogni volta che $x_j \neq 0$, y soddisfa il j-esimo vincolo con uguaglianza
2. ogni volta che $y_i \neq 0$, x soddisfa il i-esimo vincolo con uguaglianza

Allora x e y sono entrambe Ottime.

Sappiamo che una base del primale è anche una base del duale.

QUINDI:

1. Dato un vertice del problema primale, se nella sua duale la soluzione generata dalla stessa base è ammissibile, Allora la soluzione di base di quella stessa base è Ottimo in entrambi i problemi.
2. Se uno degli elementi di x_B è 0, la soluzione di base è Degenere.
3. Se tutti gli elementi di y_B sono ≥ 0 allora la soluzione di base è ammissibile (vertice).

2.6.2 Test di Ottimalità

Per sapere se un dato vertice di un (P) in forma primale è ottimo:

1. trovo la base che lo genera
2. trovo la duale di (P)
3. dal sistema $A^T y = c$ pongo a 0 le componenti Non Di Base e risolvo
4. se risolvendo le componenti Di Base tornano tutte positive, scopriamo che (la soluzione di base è ammissibile poiché rispetta tutti i vincoli $y \geq 0$ e di conseguenza) la nostra base genera un ottimo in entrambi i problemi.

2.7 Algoritmo del Simplesso

Questo è l'algoritmo usato per trovare iterativamente la soluzione ottima di un Problema di Programmazione Lineare (PL), facendo uso della Teoria della Dualità.

Dato un problema PL in primale standard: $(P) \begin{cases} max\ c^T \cdot x \\ Ax \leq b \end{cases}$

Ne trovo il duale complementare $(P) \begin{cases} min\ b^T \cdot y \\ A^T y = c \\ \forall k,\ y_k \geq 0 \end{cases}$

Parto da una soluzione di base ammissibile del primale, ovvero un Vertice.

Data la sua base B, ne calcolo la soluzione nel duale, secondo le modalità descritte nella Teoria della Dualità, se la soluzione duale è ammissibile (tutte $y_B \geq 0$) questo vertice è soluzione, altrimenti devo fare un Passo del Simplesso.

Se parto da una base con soluzione non ammissibile nel primale, che è invece soluzione ammissibile nel duale, devo fare un Passo del Simplesso Duale.

2.7.1 Passo del Simplesso Primale

1. Determino l'indice uscente h: il minore indice (di base) la cui $y_h \leq 0$. si prende sempre il minore per la Regola Anticiclo di Blend. $y_B = ((A_B)^T)^{-1}$ e $y_N = (0, 0, ..., 0)$

2. Determino $W = -A_B^{-1}$ e W^l è una colonna di W determinata dalla l-esima riga di A, in particolare considero W^h, ovvero la colonna di W determinata dalla h-esima (con h indice uscente) riga di A.

3. Calcolo per ogni indice i non di base il prodotto scalare $A_i W^h$ e scarto gli indici per cui tale rapporto è negativo

4. Rimpiazzo h con l'indice i non di base che produce il minimo del seguente prodotto:

$$r_i = \frac{b_i - A_i \overline{x}}{A_i W^h}$$

- se due indici hanno lo stesso r scelgo l'indice minore.
- se tutti gli indici i portano ad $A_i W^h \leq 0$ allora la soluzione è illimitata.

2.7.2 Passo del Simplesso Duale

In questo caso partiamo da un Problema di Programmazione Lineare (PL)in Formato Duale Standard, prendiamo un vertice, scopriamo che non è ammissibile nel primale e quindi dobbiamo applicare il Simplesso Duale.

$$(D) = \begin{cases} min\ b^T \cdot y \\ y^T A = c^T \\ y \geq 0 \end{cases}$$

1. data una base B, trovo la soluzione di base $\overline{y}_B^T = c^T A_B^{-1}$ e scopriamo che è ammissibile (tutte componenti di $\overline{y} \geq 0$)

2. Trovo quindi la complementare $\overline{x} = A_B^{-1} \cdot b_B$, se trovo che questa $\overline{x}$ non è ammissibile nel primale (non rispetta tutti i vincoli, quindi $A_N \cdot \overline{x} \geq b_N$) allora sappiamo che la nostra $\overline{y}$ non è Ottimo (fallisce il Test di Ottimalità del Teoria della DualitàTeorema della Dualità Forte / degli Scarti Complementari).

3. Allora $\exists j \in N : A_j \cdot \overline{x} > b_j$ (esiste almeno una riga violata), sia k la prima riga violata (indice minore - Regole Anticiclo di Blend), l'indice entrante.

4. Calcoliamo $W = -A_B^{-1}$

5. Calcoliamo $A_k W^i$ con $i \in B$ e elimino gli indici per i quali viene positivo (se tutti gli indici portano a prodotto positivo il minimo è $= -\infty$)

6. Costruisco i rapporti $r_i = \frac{-y_i}{A_k W^i}$ con $i \in B$ e prendo l'indice che porta al rapporto minore, e lo chiamo h, indice uscente e questo viene rimpiazzato da k (se due r sono uguali prendo l'indice minore).

2.8 Duale Ausiliario

Vediamo ora come costruire e come utilizzare il Duale Ausiliario: A partire da un problema in Problema di Programmazione Lineare (PL) in Formato Duale Standard:

$$(D) = \begin{cases} min\ y^T b \\ A^T y = c \\ y \geq 0 \end{cases}$$

costruisco il suo Duale Associato + tante epsilon quante sono le righe del duale moltiplicandole per l'identità (i)

$$(D_{aux}) = \begin{cases} min \sum_{i=0}^{n} \epsilon_i \\ y^T A + \epsilon^T = c^T \\ y \geq 0 \\ \epsilon \geq 0 \end{cases}$$

2.8.1 Utilizzo del Daux / Algoritmo di Verifica del Poliedro Vuoto

$$(D) \begin{cases} \min \quad b^T y \\ A^T y = c \\ y \geq 0 \end{cases} \quad \rightarrow \quad (D_{aux}) \begin{cases} \min \sum_i \epsilon_i \\ A^T y + \epsilon = c \\ y \geq 0 \\ \epsilon \geq 0 \end{cases}$$

soluzione di base con $B = \{0\ 1\ ...\ 2n\}$:

$$\begin{pmatrix} y \\ \epsilon \end{pmatrix} = \begin{cases} y_1 \\ y_2 \\ ... \\ y_n \end{cases} \quad \begin{pmatrix} 0 \\ 0 \\ ... \\ 0 \\ c_1 \\ c_2 \\ ... \\ c_n \end{pmatrix} \begin{cases} \epsilon_1 \\ \epsilon_2 \\ ... \\ \epsilon_n \end{cases}$$

Il duale ausiliario (Daux) viene utilizzato per trovare una base ammissibile in un problema in forma duale std (D).

Una volta costruito il duale ausiliario, la base costituita dai primi 2n vincoli (con n dimensione del vettore y) è una base la cui soluzione di base è ammissibile (soluzione: tutte le y a 0 e tutte le epsilon uguali alle relative c).

Da questa soluzione di base possiamo applicare il simplesso Duale fino all'ottimo.

- Se il valore ottimo di (Daux) è maggiore di zero allora (D) non ha soluzioni ammissibili.

- Se il valore ottimo di (Daux) è uguale a zero, allora (D) ha almeno una soluzione di base ammissibile, che si può costruire a partire da una base ottima per (Daux).

ATTENZIONE: prima di costruire (Daux) è necessario fare in modo che tutte le c in (D) siano ≥ 0, semplicemente cambiando semmai di segno alle equazioni.

Altrimenti la soluzione di base di (Daux) indicata dai primi 2n indici non sarebbe ammissibile.

2.9 Problema di Produzione

$$\begin{cases} \max c^t \cdot x \\ A \cdot x \leq b \end{cases}$$

Partiamo con un esempio:

Una ditta produce laminato di Tipo A e laminato di Tipo B (il prodotto).
Ogni laminato passa per i reparti MateriePrime, Taglio, Finiture Tipo A, Finiture Tipo

B (in base a se è tipo A o B).
Il guadagno è 8,4 per il Tipo A e 11,2 per il tipo B.
I vincoli sono:
Tipo A deve stare 30h in MateriePrime, 10h in Tagli, 20h in Finiture A.
Tipo B deve stare 20h in MateriePrime, 20h in Tagli, 30h in Finiture B.
Materie prime può fare massimo 120h, Tagli 80h, Finiture A 62h, Finiture B 105h.
Dobbiamo trovare la combinazione di Tipo A e Tipo B che massimizzi il guadagno, ovviamente rimanendo entro i vincoli.

Troviamo ora la matrice A e i vettori c, b:

$$A = \begin{bmatrix} 30 & 20 \\ 10 & 20 \\ 10 & 0 \\ 0 & 30 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$c = \begin{bmatrix} 8.4 \\ 11.2 \end{bmatrix}$$

$$b = \begin{bmatrix} 120 \\ 80 \\ 62 \\ 105 \\ 0 \\ 0 \end{bmatrix}$$

L'intersezione dei semipiani dati da $A \cdot x \leq b$ è la **Regione Ammissibile**[1]. Adesso siamo in grado di risolvere l'esercizio tramite l'Algoritmo del Simplesso o trovare direttamente un valore ottimo tramite Matlab Linprog.

[1]La regione ammissibile è l'insieme di valori ammissibili come soluzioni per un problema. L'ammissibilità viene valutata in base ai vincoli.

3 Programmazione Lineare Intera (PLI)

Questa classe di problemi richiede che la Regione Ammissibile sia composta da punti a componenti intere. Dovremmo perció spesso aggiungere il vincolo di interezza: $x \in Z_+^n$

La tecnica di risoluzione di questi problemi si fonda sulla diminuzione del gap tra V_I e V_S tramite *algoritmi di riduzione del gap.*

3.1 Algoritmo di Riduzione del Gap tramite Piano di Taglio

Questo algoritmo di Riduzione del Gap può essere usato per risolvere qualsiasi Problema di Programmazione Lineare Intera (PLI).
Si basa sul concetto di albero di enumerazione totale (vd. Figure 1).
Consiste nel Calcolare un piano di taglio, ovvero un vincolo, e aggiungerlo al problema di partenza, iterativamente, fino a quando non si ottiene un errore nel target.

Algoritmo:
1. Calcolo X_{RC}
2. Porto il problema in Problema di Programmazione Lineare (PL) Formato Duale Standard
3. Ricalcolo X_{RC} nel duale, ovvero calcolo gli scarti
4. Trovo $B =$ (indici di componenti $\neq 0$) e $r =$ (primo indice di componente frazionaria)
5. $\tilde{A} = A_B^{-1} \cdot A_N$, nomino le righe di $\tilde{A}$ con gli stessi indici di A_B
6. Scrivo il piano di Taglio:

$$r: \quad (.. \{\tilde{A}_r\} ..) \cdot \begin{pmatrix} .. \\ X_N \\ .. \end{pmatrix} \geq \{X_r\}$$

dove $\{\}$ indica parte frazionaria e si calcola : $\{C\} = C - \lfloor C \rfloor$ ovvero = numero - arrotondamento per difetto, che equivale all'aritmetica modulo 1.
7. se necessario possiamo scrivere il piano di taglio con le variabili del primale, ovvero sostituire lo scarto con lo scarto in funzione delle variabili del primale.

3.2 Algoritmo di Riduzione del Gap Branch and Bound

Parto con V_i come valore corrente.
Tramite le regole di taglio:
Taglio un ramo e scopro se tra le soluzioni sotto quel ramo ce ne è una migliore del valore corrente, in caso contrario taglio via quel ramo e quindi nego la variabile di quel ramo.
Se invece trovo un valore migliore, lo tengo come valore corrente, taglio il ramo e fisso la variabile di quel ramo.
Tagliare: visita implicita del sottoalbero, ovvero ho tutte le informazioni che mi servono su tutte le foglie del sottoalbero.
Questo algoritmo finisce quando ho fatto una visita implicita per ogni ramo.

3.2.1 Regole di Taglio per problemi di Minimo

con P_{ij} stadio i, ramo j.
1. $P_{ij} = \emptyset \quad ? \quad taglio$

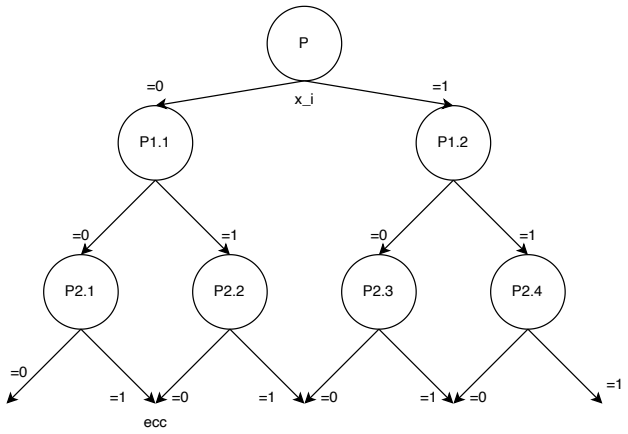

Figure 1: Rappresentazione dell'Albero di Enumerazione Totale

devo controllare poiché più scendo e più restringo la Regione Ammissibile (aggiungo vincoli).
2. $V_I(P_{ij}) \geq V_S(P) \quad ? \quad taglio$
Calcolo $V_I(P_{ij})$, ovvero l'ottimo del Rilassato continuo di P_{ij}, se è maggiore del valore corrente posso tagliare.
3. se $V_I(P_{ij}) < V_S(P) \quad$ e $\quad V_I(P_{ij})\ ammissibile \quad$ allora aggiorno V_S con $V_I(P_{ij})$

3.2.2 Regole di Taglio per problemi di Massimo

1. $P_{ij} = \emptyset \quad ? \quad taglio$
2. $V_S(P_{ij}) \leq V_I(P) \quad ? \quad taglio$
Calcolo $V_S(P_{ij})$, ovvero l'ottimo del Rilassato continuo di P_{ij}, se è minore o uguale del valore corrente posso tagliare.
3. se $V_S(P_{ij}) > V_I(P) \quad$ e $\quad V_S(P_{ij})\ ammissibile (\in Z^n$ ovvero componenti intere) allora aggiorno V_I con $V_S(P_{ij})$ allora scatta la regola 2 e taglio.

3.3 Teorema Astratto di Equivalenza tra PL e PLI

Prendiamo un poliedro P, ne prendiamo i solu punti a componenti intere e chiamiamo questo insieme di punti S. Adesso consideriamo $convS$, ovvero il più piccolo poliedro che comprende tutti i punti di S, che si ottiene "connettendo" i vertici di S.

dati $A, b \in Z$

$$max_{x \in S}\ c^T x = max_{x \in convS}\ c^T x$$

questo perché i vertici di questi due insiemi sono gli stessi! E sappiamo che l'ottimo sta sui vertici.

Possiamo considerare S la regione ammissibile di un problema PLI e $convS$ la regione ammissibile di un problema PL.

3.4 Problema di Trasporto

con:
- c_{ij} costo da i a j
- o_i offerta di i
- d_j domanda di j

$$\begin{cases} \min \sum_i \sum_j c_{ij} \cdot x_{ij} \\ \sum_j x_{ij} = o_i \quad \forall i \\ \sum_i x_{ij} = d_j \quad \forall j \\ x_{ij} \geq 0 \end{cases}$$

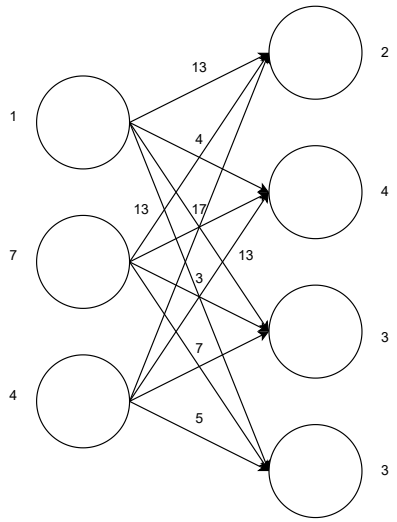

Figure 2: Rappresentazione dei Fornitori e dei clienti e dei relativi archi(collegamenti)

Questo Problema è un caso particolare Problema di Programmazione Lineare su Reti non Capacitate (che vedremo più avanti) e pertanto gode del Teorema di Interezza, per ora ci basta sapere che se i dati forniti sono interi allora anche la (le) soluzione ottima è a componenti intere.
Questo problema consiste nel trovare il minimo costo possibile di trasporto di oggetti, dati diversi stabilimenti di spedizione e diversi stabilimenti di arrivo.
Ogni stabilimento di spedizione ha un costo diverso per spedire ad uno stabilimento di arrivo.
I vincoli consistono nella disponibilità degli stabilimenti di spedizione e nella domanda degli stabilimenti di arrivo.
Se l'offerta è maggiore della domanda una soluzione possibile è lo stoccaggio, se invece la domanda è maggiore dell'offerta il Poliedro[2] è vuoto. Per la risoluzione si utilizza l'Algoritmo del Simplesso.

[2]Un Poliedro in geometria è una intersezione di un numero finito di semispazi lineari e chiusi.

3.5 Problema di Assegnamento

Questo problema è a sua volta caso particolare del problema di Trasporto, e pertanto gode a sua volta del teorema di interezza.
Consiste nell'avere per esempio n ingegneri ed n progetti da realizzare, ogni ingegnere può realizzare al massimo un progetto ed ogni progetto può essere realizzato da un solo ingegnere. Ogni ingegnere impiega un numero di risorse diverso per realizzare ogni progetto e lo scopo è completare tutti i progetti nel minor impiego di risorse possibile.
Questo problema può essere non cooperativo ($x_{ij} \in \{0,1\}$), o cooperativo ($x_{ij} \in [0,1]$).

$$\begin{cases} \min \sum_i \sum_j c_{ij} \cdot x_{ij} \\ \sum_i x_{ij} = 1 \quad \forall j \in N \\ \sum_j x_{ij} = 1 \quad \forall i \in N \\ x_{ij} \in \{0,1\} \end{cases}$$

3.6 Problema dello Zaino

Questa famiglia di problemi appartiene a quella dei Problema di Programmazione Lineare (PL). Consiste nell'avere vari oggetti da trasportare, ognuno con un "peso" p (o ingombro) diverso e un valore v, ed uno "zaino", con una certa capienza C. Il problema consiste nel trovare la combinazione di oggetti che permette di portare più peso possibile e quindi "saturare" lo zaino.

$$\begin{cases} \max v^T \cdot x \\ \sum_i x_i \leq C \\ x_i \in \{0,1\} \quad \text{oppure} \quad x_i \in N \end{cases}$$

- binario: ogni oggetto può essere portato oppure no, una sola volta: x può essere $0, 1$
- intero: ogni oggetto può essere preso quante volte si vuole: x può essere preso n volte ($n \in N$)

3.7 Problema del Bin-Packing

Questo è un Problema di Programmazione Lineare Intera (PLI). Ho un numero di oggetti di cui conosco il peso e ho un indeterminato numero di contenitori (Bin) di cui conosco la capienza. Trovare l'assegnamento che porti all'utilizzo del minor numero possibile di contenitori.

n oggetti, m contenitori
Variabili:
- $x_{ij} = (j \text{ è dentro} i) \quad ? \quad 1 : 0$
- $y_i = (i \text{ è usato}) \quad ? \quad 1 : 0$

$$\begin{cases} \min \sum_{i=1}^{m} y_i \\ \sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j \in [i; n] \\ \sum_{j=1}^{n} p_j x_{ij} \leq C y_i \quad \forall i \in [i; m] \\ x_{ij} \in \{0,1\} \\ y_i \in \{0,1\} \end{cases}$$

A è di dimensioni: $(n+m) \times (n \cdot m + m)$

3.7.1 Risoluzione

Essendo questo un problema di PLI, troviamo una V_S, una V_I, e applichiamo un algoritmo di riduzione del gap.
Il più adatto, per la natura binaria del problema, è l'Algoritmo di Riduzione del Gap Branch and Bound. Abbiamo bisogno di una approssimazione del numero di bin necessari, meno ne mettiamo nel modello e più velocizziamo. Alla peggio dovremo utilizzare n contenitori (numero di oggetti).

3.7.2 Algoritmi per la Vs del Bin-Packing

Per trovare la soluzione ottimale abbiamo 3 algoritmi, in ordine di accuratezza:
decreasing = ordinare oggetti per peso decrescente

- next-fit-decreasing:
  Il primo contenitore é il contenitore corrente. Se possibile, assegna un oggetto al contenitore corrente; altrimenti assegnalo ad un nuovo contenitore, che diventa quello corrente.

- first-fit-decreasing:
  Assegna ogni oggetto al primo contenitore usato che puó contenerlo. Se nessuno di essi puó contenerlo, assegna l'oggetto ad un nuovo contenitore.

- best-fit-decreasing:
  ordino i bin per capienza rimanente decrescente prendo un oggetto, lo metto nel bin con minore capienza rimanente, se non entra, in quello dopo ripeto: riordino i bin, metto un oggetto, riordino ecc

3.7.3 Vi per Bin Packing

La valutazione inferiore di questo problema è particolarmente semplice: è l'approssimazione per eccesso del valore associato alla soluzione del rilassato continuo del problema:

$$V_I = \left\lceil \frac{\sum_{j=1}^{n} p_j}{C} \right\rceil$$

3.8 Problema del Commesso Viaggiatore (TSP)

TSP = traveling salesman problem = problema del commesso viaggiatore

Dati dei nodi collegati da archi, con ogni arco con relativo costo:
Trovare il cammino di costo minimo che includa tutti i nodi una ed una sola volta, detto Ciclo Hamiltoniano. Questo problema su n nodi ha $n!$ soluzioni ammissibili.
Può essere simmetrico (gli archi possono essere attraversati in entrambi i versi allo stesso costo) oppure asimmetrico (gli archi possono essere attraversati in uno solo dei versi, indicati dal verso della freccia).

3.8.1 Modello Asimmetrico

N nodi
Variabili: $x_{ij} = (arco\ (i,j) \in ciclo) \quad ? \quad 1 : 0$

$$\begin{cases} \min \sum_i \sum_j c_{ij} \cdot x_{ij} \\ \sum_j x_{ij} = 1 \quad \forall i \\ \sum_i x_{ij} = 1 \quad \forall j \\ \sum_{\substack{i \in S \\ j \notin S}} x_{ij} \geq 1 \quad \forall S \in N, \quad S \neq \emptyset, S \neq N \\ x_{ij} \in \{0,1\} \end{cases}$$

3.8.2 Modello Simmetrico

Potremmo usare le tecniche risolutive del TSP Asimmetrico ma come segue possiamo dimezzare le variabili.

$$\begin{cases} min\ \sum_i \sum_{j>i} c_{ij} \cdot x_{ij} \\ \sum_{i<j} x_{ij} + \sum_{j<i} x_{ji} = 2 \forall j \\ \sum_{\substack{i \in S \\ j \notin S}} x_{ij} + \sum_{\substack{i \notin S \\ j \in S}} x_{ji} \geq 1 \quad \forall S \subset N, \quad S \neq \emptyset, S \neq N \\ x_{ij} \in \{0,1\} \end{cases}$$

I Primi si chiamano vincoli di grado, ovvero ogni nodo ha tot legami, grado 2, e sono n vincoli (con $n = |nodi|$). In ogni vincolo ci sono $n-1$ legami.
I secondi si chiamano vincoli di connessione.
Le variabili sono $|x| = \frac{n!}{|S|!(n-|S|)!}$ La matrice Aeq avrà dimensioni $n \times |x|$
La matrice A avrà dimensioni $(2^n - 2n - 2) \times |x|$ ma in realtà si dimezzano.

3.8.3 Risoluzione

- V_I: Trovare k-Albero tramite Algoritmo di Kruskal [3].
- V_S: Algoritmo del Nodo più vicino.
- Riduzione del gap: Algoritmo di Riduzione del Gap Branch and Bound.

3.9 Problema di Copertura

Supponiamo di dover procurare un servizio. Costruiamo la matrice di Copertura che indica se un utente è servito o meno dalle varie postazioni. Dobbiamo minimizzare i costi, la somma dei costi di apertura delle postazioni.
È un Problema di Programmazione Lineare Intera (PLI) NP-HARD.

[3] isolo il nodo k, collego i $n-1$ rimanenti nodi con i $n-2$ archi meno costosi ponendo attenzione a non creare cicli, infine collego il nodo k al resto della struttura tramite i due archi meno costosi, ottengo così un k-albero, ovvero una struttura connesa con un elemento di grado 2

- I=insieme siti candidati ad ospitare il servizio - J=insieme di nodi domanda
- d_{ij} = distanza i, j
- c_i = costo di aprire servizio in i
- D = distanza massima per considerare domanda coperta da servizio
- minimizzare i costi

$$\begin{cases} \min \sum_i c_i \cdot x_i \\ \sum_i a_{ij} \cdot x_i \geq 1 \quad \forall j \quad a_{ij} = (d_{ij} \leq D) \ ? \ 1 : 0 \\ x_i \in \{0, 1\} \end{cases}$$

3.9.1 Regole di Riduzione di una Matrice a Priori

1. una riga di tutti 0 vuol dire Regione Ammissibile vuota, quindi non si può presentare
2. una colonna di tutti 0 possiamo toglierla
3. una riga di tutti 1 possiamo toglierla
4. una riga con un solo 1 vuol dire che devo porre a 1 la variabile della postazione che lo serve, posso poi quindi cancellare tutti i quartieri serviti dalla postazione
5. regola di dominanza: si applica alle colonne (postazioni), se una postazione copre tutti i quartieri (o più) che copre un'altra postazione, posso eliminare la colonna di quest'ultima. $A^k \geq A^r$, regola applicabile solo in mancanza di costi.

3.9.2 Risoluzione

- V_I: Risolvo il rilassato continuo $\begin{cases} min \quad c^T \cdot x \\ Ax \geq \vec{e} \\ 0 \leq x \leq 1 \end{cases}$ e arrotondo per eccesso essendo un problema di minimo
- V_S: una sluzione ammissibile

  1. senza costi: (caso particolare dell'algoritmo con costi) saturiamo progressivamente, apriamo postazioni con più quartieri forniti, cancello la colonna della postazione e cancello le righe servite, vediamo cosa rimane e ripetiamo

  2. con i costi: Algoritmo di Chvatal[4]

- Algoritmo di riduzione del gap: Algoritmo di Riduzione del Gap tramite Piano di Taglio.

3.10 Problema di Massima Copertura

Associo ad ogni utente/quartiere il numero di clienti/abitanti stimati, con un dato numero di postazioni da poter aprire, devo massimizzare il numero di clienti serviti dalle postazioni.

[4] 1. Ordino le postazioni per il loro costo unitario, ovvero il costo di apertura diviso per la somma degli utenti coperti.
2. Apro la postazione con costo unitario minimo, ne cancello la colonna e ne cancello le righe che serve e depenno la postazione. Inoltre cancello le righe di tutti 1 se si presenta.
3. Se ho ancora righe: Ricalcolo i costi unitari, ovvero il costo di apertura diviso per la somma degli utenti, non depennati, coperti. Torno al punto '2'.

- I = insieme siti candidati ad ospitare il servizio
- J = insieme nodi domanda
- h_j = domanda associata al nodo j
- p = numero prefissato di postazioni da aprire
- d_{ij} = distanza tra i e j
- D = distanza di copertura
- $z_j = 1$ se il nodo j è coperto dalla soluzione scelta
- $a_{ij} = 1$ se il nodo j è coperto da i

$$\begin{cases} \max \sum_j h_j \cdot z_j \\ \sum_i a_{ij} \cdot x_i \geq z_i \quad \forall j \quad (1) \\ \sum_i x_i = p \\ x_{ij} \in \{0,1\} \\ z_j \in \{0,1\} \end{cases}$$

(1): per coprire il nodo j, dobbiamo aprire il servizio in almeno una postazione che lo copra.

4 Programmazione Matematica su Reti (PLR)

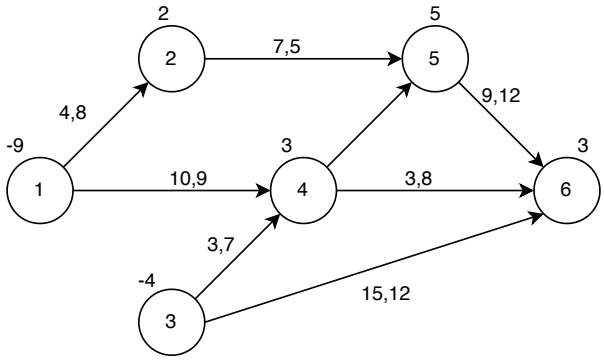

Figure 3: Rappresentazione di una rete con bilanci ai nodi e (in ordine) costo e capacità degli archi

In questa classe di problemi abbiamo n nodi collegati da vari archi, ogni nodo ha un **bilancio**, ovvero il numero di unità che richiede (se è negativo il nodo è una sorgente, altrimenti è un pozzo) e ogni arco ha costo (unitario) e capacità.

4.1 Matrice di Incidenza su Reti

Si usa per sostituire un disegno della rete. Ha tante righe quanti sono i nodi. Ha tante colonne quanti sono gli archi.
Per ogni casella:
- se il nodo è sorgente dell'arco, si mette -1
- se il nodo è destinazione dell'arco, si mette 1
- se il nodo non ha a che fare con l'arco, si mette 0

Ovvero per l'arco a_{ij}, su i si mette -1, su j si mette 1, sugli altri si mette 0. Su ogni colonna ci sono esattamente UN -1 ed UN 1.
Questa matrice non è di Rango massimo, le righe non sono linearmente indipendenti, d'altronde se conoscono i bilanci su $n-1$ nodi, posso facilmente ricavare il bilancio al n-esimo nodo, poiché la somma deve essere pari a *zero*. Posso quindi eliminare una riga della matrice.

4.2 Teorema di Interezza

I vertici di un poliedro $\begin{cases} Ex = b \\ x \geq 0 \end{cases}$ sono a componenti intere, con E matrice di incidenza su reti

Una matrice di incidenza su reti è una matrice (di dimensioni #(nodi) x #(archi)) che presenta su ogni colonna degli 0, UN +1 ed UN -1. Un albero di copertura sulla rete è descritto da una sottomatrice di E quadrata di dimensioni (n-1)x(n-1). Questa sottomatrice è quindi a sua volta composta da 0,1 e -1, e con opportune permutazioni di riga e/o colonna, può essere portata in forma triangolare inferiore con sulla diagonale +1 e -1. Una matrice triangolare inferiore con 1 e -1 sulla diagonale ha determinante +-1. L'indeterminatezza del segno deriva dal fatto che le permutazioni di riga e di colonna

cambiano il segno del determinante. Questo tipo di matrice è unimodulare, e le soluzioni di sistemi lineari su queste matrici producono risultati a componenti intere, per il teorema di Cramer Ne otteniamo che le soluzioni di sistemi Ex=b con E matrice di incidenza e b a componenti intere (NOTA BENE!) sono a loro volta a componenti intere.
Questo Teorema vale sui problemi su reti e sui loro derivati: trasporto, assegnamento ecc.

4.3 Teorema di Caratterizzazione delle Basi

Ipotesi:
La matrice E (unimodulare: $det(E) = 1$ o -1) ha rango $n-1$ e gli alberi di copertura hanno $det \neq 0$ (quindi in realtà $dim(E) = (n-1) * m$)

Tesi:
Tutte le basi (T) sono alberi

Questo Teorema applicato alla Programmazione Matematica a Reti porta al seguente risultato:
Sia T, L, U una tripartizione degli archi della rete con T albero di copertura $\implies$ scegliendo T, U, T', L' ottengo una base.
Vale anche l'inverso, data una base si ottiene da T, U, T', L'.

4.4 Problema del Flusso di Costo Minimo

$$\begin{cases} min \quad c^T \cdot x \\ E \cdot x = b \\ 0 \leq x_{ij} \leq u_{ij} \end{cases}$$

dove b sono i bilanci ai nodi oppure

$$\begin{cases} \min \sum_{(i,j)\in A} c_{ij} \cdot x_{ij} \\ \sum_i x_{ij} - \sum_i x_{ji} = b_j \quad \forall j \quad (1) \\ l_{ij} \leq x_{ij} \leq u_{ij} \quad \forall (i,j) \in A \quad (2) \end{cases}$$

dove A è l'insieme degli archi, l è la capacità minima, u è la capacità massima, (1) sono i vincoli di bilancio ai nodi e (2) sono i vincoli di capacità.

Per definire un base nelle reti capacitate dobbiamo prima modificare leggermente il modello introducendo una variabile w_{ij} che indica la **portata residua** di un arco:

$$\begin{cases} min \quad C^T \cdot x \\ E \ x = b \\ x_{ij} + w_{ij} = u_{ij} \\ x \geq 0 \\ w \geq 0 \end{cases}$$

ovvero in forma matriciale, duale standard:

$$\begin{cases} \min \quad \begin{pmatrix} c \\ 0 \end{pmatrix}^T \begin{pmatrix} x \\ w \end{pmatrix} \\ \begin{pmatrix} E^T & I \\ 0 & I \end{pmatrix}^T \begin{pmatrix} x \\ w \end{pmatrix} = \begin{pmatrix} b \\ u \end{pmatrix} \\ (x, w) \geq 0 \end{cases}$$

Con $M = \begin{pmatrix} E^T & I \\ I & I \end{pmatrix}$ di dimensione $2m \times (n-1+m)$ e rango $n-1+m$.

4.5 Teorema di Bellman

Dato un albero di copertura T, che genera un flusso di base (x_T) ammissibile, se sono rispettate le ***Condizioni di Bellman*** *allora siamo all'ottimo.*

Condizioni di Bellman:
Definito il costo ridotto relativo ad un arco:

$$C_{ij}^{\pi} = c_{ij} + \pi_i - \pi_j$$

- $C_{ij}^{\pi} = 0 : (i,j) \in T$ ovvero costi ridotti di archi di base $= 0$
- $C_{ij}^{\pi} \geq 0 : (i,j) \in L$ ovvero costi ridotti di archi $\in L$ (non di base) ≥ 0
- $C_{ij}^{\pi} \leq 0 : (i,j) \in U$ ovvero costi ridotti di archi $\in U$ (non di base) ≤ 0

4.6 Algoritmo del Simplesso su Reti

1. Troviamo un vertice di partenza $T \rightarrow x_T \geq 0$ ammissibile con potenziale non ammissibile (Teorema di Bellman), ovvero con almeno un costo ridotto relativo ad un arco non di base minore stretto di zero. Scelgo il primo vincolo che viola Bellman: $(i,j) \in L : c_{ij}^{\pi} < 0$, e questo è l' arco entrante.

2. Disegno l'albero di copertura (solo archi di base) e aggiungo tratteggiato l'arco entrante. Questo creerà un ciclo all'interno della Rete. Adesso devo trovare l'arco uscente, in particolare l'arco uscente deve far parte del ciclo, in modo da "spezzare" il ciclo.

   (a) orientiamo il ciclo in maniera concorde con (i,j), l'arco entrante

   (b) $C = C^+ \cup C^-$ dove C^+ sono gli archi concordi con il verso del ciclo, e C^- quelli discordi.

   (c) Regola di Update: con $\theta \geq 0, \in N$

$$x(\theta) = \begin{cases} \overline{x}_{ij} + \theta & (i,j) \in C^+ \\ \overline{x}_{ij} - \theta & (i,j) \in C^- \\ \overline{x}_{ij} & (i,j) \notin C \end{cases}$$

   Teorema: La funzione obiettivo, calcolata sul nuovo flusso, è uguale al costo precedente, più theta volte il costo dell'arco entrante:

$$c^T x(\theta) = c^T \overline{x} + \theta \cdot c_{ij}^{\pi}$$

3. Prendo $\theta = min\{\overline{x}_{ij}\}$ $(i,j) \in C$ e il primo arco di cui prendo il valore è l'arco uscente.

4.6.1 Variazione Per Reti Capacitate

Il simplesso su Reti capacitate è pressoché identico a quello su reti non capacitate (da notare che sono diverse le condizioni di Bellman. La differenza è la seguente:
Se l'arco (ij) entrante, ovvero il primo arco in ordine lessico-grafico che viola Bellman è appartenente ad U invece che a L oriento il ciclo in maniera discorde all'arco entrante; calcolo:

- $\theta^+ = \min\{u_{ij} - x_{ij} : (ij) \in C^+\}$
- $\theta^- = \min\{x_{ij} : (ij) \in C^-\}$ se $C^+ = \emptyset$ allora $\theta^+ = +\infty$ e altrettanto per C^- e θ^-
- θ è quindi il minore tra θ^+ e θ^- e l'arco (ij) relativo a θ è l'arco uscente.

Se l'arco uscente era in C^+, esce da T ed entra in U, se invece era in C^-, esce da T ed entra in L

4.7 Problema dei Potenziali su Reti

Il problema dei potenziali su rete è il Duale della forma matriciale duale standard del Problema del Flusso di Costo Minimo: $(flusso\ minimo)$

$$\begin{cases} \min \quad \begin{pmatrix} c \\ 0 \end{pmatrix}^T \begin{pmatrix} x \\ w \end{pmatrix} \\ \begin{pmatrix} E^T & I \\ 0 & I \end{pmatrix}^T \begin{pmatrix} x \\ w \end{pmatrix} = \begin{pmatrix} b \\ u \end{pmatrix} \\ (x,w) \geq 0 \end{cases}$$

ovvero

$$\begin{cases} \max \quad (b \quad u)^T \begin{pmatrix} \pi \\ \mu \end{pmatrix} \\ \begin{pmatrix} E^T & I \\ 0 & I \end{pmatrix} \begin{pmatrix} \pi \\ \mu \end{pmatrix} \leq \begin{pmatrix} c \\ 0 \end{pmatrix} \end{cases}$$

che equivale a:

$$\begin{cases} \max \quad \pi^T b + \mu^T u \\ \pi^T E + \mu^T \leq c^T \\ \mu \leq 0 \end{cases}$$

Il potenziale di base $\overline{\pi}$ è ammissibile se e solo se i costi ridotti ($c^{\pi}_{ij} = \pi_i + c_{ij} - \pi_j$)degli archi in L sono non negativi ed i costi ridotti degli archi in U sono non positivi.

4.8 Problema del Cammino di Costo Minimo

Un cammino dal nodo n_1 al nodo n_{p+1} è un insieme di archi $a_1, ..., a_p$ tale che per ogni $i = 1, ..., p$ si ha che $a_i = (n_i, n_{i+1})$ oppure $a_i = (n_{i+1}, n_i)$ ed inoltre che gli archi siano tutti diversi tra loro. Se vale sempre la prima condizione, cioè gli archi coinvolti vanno dal nodo n_i al nodo n_{i+1}, il cammino si dice orientato. Questo è il problema di trovare (se esistono) i cammini orientati di costo minimo da un nodo r verso tutti gli altri nodi. Tale problema si può formulare come un problema di flusso di costo minimo ponendo le capacità superiori $u_{ij} = +\infty$ ed i bilanci dei nodi tutti uguali ad 1 tranne quello del nodo r:

$$b = \begin{cases} 1 & se\ i \neq r \\ -n+1 & se\ i = r \end{cases}$$

$$\begin{cases} \min \sum_{(i,j)\in A} c_{ij} \cdot x_{ij} \\ \sum_i x_{ij} - \sum_i x_{ji} = b_j \quad \forall j \quad b = \begin{cases} 1 & se\ i \neq r \\ -n+1 & se\ i = r \end{cases} \\ x_{ij} \in N \quad \forall (i,j) \in A \end{cases}$$

Questo problema si risolve attraverso l'algoritmo di Dijkstra.

4.9 Problema del Flusso di Costo Massimo

Questo è un Problema di Programmazione Matematica a Reti Capacitate (PLRC) con ogni arco di costo nullo, in cui si chiede di trovare il flusso massimo tra due nodi. Per convenzione chiamiamo il nodo 1, s (source) ed il nodo n, t (tail). È possibile vedere il problema in maniera leggermente diversa in modo da semplificare l'apprendimento, in particolare aggiungiamo un arco, l'arco (n, s) di costo -1, che quindi porta ad un guadagno per ogni unità di flusso che lo attraversa; Il problema diventa la massimizzazione del flusso su questo ultimo arco, in particolare chiamiamo questo flusso $x_{n,1}$, v, che sarà la soluzione ottima del nostro problema. Viene risolto tramite l'Algoritmo di Ford Fulkerson Edmond Karp (FFEK).

$$\begin{cases} \max \quad v \\ Ex = b \quad con \quad b = \begin{cases} -v & i = s \\ 0 & i \neq s, t \\ v & i = t \end{cases} \\ 0 \leq x_{(i,j)} \leq u_{(i,j)} \end{cases}$$

ovviamente è possibile scrivere la funzione obiettivo come $-\min \quad -v$.

4.9.1 Teorema del Max Flow - Min cut

$$\max\{\overline{x}\} = \min\{\ u(N_s; N_t) \quad \forall \quad (N_s; N_t)\ \}$$

: Il Flusso Massimo è pari alla portata del taglio di portata minima.
Il Problema del Taglio di Portata minima è il problema duale del problema di Flusso Massimo.

5 Programmazione non Lineare

5.1 Note Introduttive di Analisi II

- il gradiente è il vettore composto dalle derivate parziali della funzione.
- Teorema di Fermat: con $f \in C^1$, se $\nabla f(\overline{x}) = 0$ allora $\overline{x}$ è punto di min/max locale.
- L'Hessiana è la matrice composta dalle derivate parziali (derivate due volte) della funzione:

$$Hf(x) = \begin{bmatrix} \frac{\delta^2 f(x)}{\delta x_1^2} & \dots & \frac{\delta^2 f(x)}{\delta x_1 \delta x_n} \\ \dots & & \dots \\ \frac{\delta^2 f(x)}{\delta x_2 \delta x_1} & \dots & \frac{\delta^2 f(x)}{\delta x_n^2} \end{bmatrix}$$

  Per il Teorema di Schwartz l'Hessiana è simmetrica.
- Sia $\overline{x}$ stazionario, se $Hf(\overline{x}) > 0$ allora $\overline{x}$ è minimo locale. $\implies$ se $\overline{x}$ è minimo locale, allora $Hf(\overline{x}) > 0$
- Una matrice è definita positiva (> 0) se $< Ax, x >> 0 \forall x$, è invece semidefinita positiva (≥ 0) se $< Ax, x > \geq 0 \forall x$
- Teorema: una matrice simmetrica è definita positiva se e solo se i suoi autovalori sono strettamente positivi, è semidefinita positiva se i suoi autovalori sono ≥ 0.
- Teorema: le matrici simmetriche hanno autovalori Reali. quindi sulla Hessiana posso controllare $\det(H - \lambda I) = 0, \quad \lambda > 0$?
- Quindi in questo ordine ho questi insiemi (Diagramma di Venn): Punti stazionari ($\nabla f = 0$) $\rightarrow$ $H \geq 0$ $\rightarrow$ minimi locali $\rightarrow$ $H > 0$, quindi come criterio di stop dobbiamo usare la condizione necessaria $H \geq 0$
- Teorema del calcolo della derivata direzionale:

$$f'(\overline{x}, \overline{d}) =< \nabla f(\overline{x}), \overline{d} >$$

- Forma quadratica: $f(x) = x^T Q x + c^T x$ con $2Q = Hf$ quindi se $f(x) = 6x_1^2 + 3x_1x_2 + 5x_2^2$ allora $Q = \begin{bmatrix} 6 & \frac{3}{2} \\ \frac{3}{2} & 5 \end{bmatrix}$ e $H = \begin{bmatrix} 12 & 3 \\ 3 & 10 \end{bmatrix}$
- Funzioni convesse: $f : R^n \rightarrow R$, una f si dice convessa se $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \forall \lambda \in [0,1], \forall x \in R^n$ Anche: una f si dice convessa se il gradiente è monotono crescente ($\nabla f(x+h) > \nabla f(x) \forall h > 0$), quindi: $f \in C^2$ è convessa se e solo se $Hf(x) \geq 0 \forall x$, ma il gradiente in una quadratica è costante, quindi basta calcolarsi gli autovalori.
- Teorema: se f è convessa e $\nabla f(\overline{x}) = 0$ allora $\overline{x}$ è minimo globale, poiché una f convessa non ha minimi locali ne selle.
- Funzioni coercive continue: una funzione è coerciva se $\lim_{||x|| \rightarrow \infty} f(x) = +\infty$ ovvero se va a più infinito in tutte le direzioni. Se una funzione è coerciva allora esiste un Minimo Globale (anticoerciva $\rightarrow -\infty$, esiste MG).

5.2 Problema di Programmazione non Lineare non Vincolato

Un problema PNR non vincolato è la semplice massimizzazione/minimizzazione di una funzione non lineare. I metodi di risoluzione sono algoritmi iterativi che iniziano da un punto di partenza e convergono a punti stazionari. Punto di accumulazione: punto al cui l'algoritmo si avvicina per infiniti passi. Gli algoritmi costruiscono il nuovo punto nel seguente modo:

$$x^{k+1} = x^k + t_k d^k$$

con d^k direzione e t_k step size.
Differiscono nel modo in cui si calcolano direzione e step size. Si restringe il problema ad una semiretta funzione di una variabile:

$$\Phi(t) = f(x^k + t_k d^k)$$

Per definizione d^k deve essere una direzione buona, quindi: (PROB DI MINIMO)

$$\Phi(t) = \nabla f(x^k + t_k d^k) \cdot d^k$$

scelgo $d^k = -\nabla f(x^k)$

5.2.1 Considerazioni sul numero di passi

Purtroppo in un problema non lineare non vincolato i passi possono essere illimitati, ci servono quindi delle regole di Stop, qua abbiamo alcuni esempi:

- dopo un certo numero di step/tempo
- quando il miglioramento diventa trascurabile: $|f(x^{k+1}) - f(x^k)| < 10^{-4}$ per esempio
- quando il gradiente diventa quasi zero: $||\nabla f(x^{k+1})|| < 10^{-4}$

5.2.2 Metodo del Gradiente Libero

Riassumiamo la risoluzione di un problema di minimo non lineare non vincolato tramite il metodo del gradiente:
Ipotesi: $f : R^n \to R, \quad f \in C^1$
$x^{k+1} = x^k + t_k d^k$, calcolo $d^k = -\nabla f(x^k)$ e calcolo $t_k = argmin_{t \geq 0} \Phi(t)$ con $\Phi(t) = f(x^k + t_k d^k)$

5.2.3 Metodo del Passo costante

per MAX

$$d^k = \nabla f(x^k)$$
$$0 \leq t_k \leq \frac{2}{L} \quad con\ L \geq |\frac{\sigma^2 f}{\sigma x_1 x_2}|$$

5.2.4 Metodo di Newton

$$d^k = H \cdot f(x^k)^{-}1 \cdot \nabla f(x^k)$$
$$t_k = 1$$

5.2.5 Teorema del Metodo del Gradiente

La successione del gradiente con ricerca esatta, o termina in un numero finito di passo in un punto stazionario, oppure i suoi punti di accumulazione sono punti stazionari.

5.3 Problema di Programmazione non Lineare Vincolato

5.3.1 Definizione di Dominio PNL

$f : D \subset R^n \to R$

$$D = \{x \in R^n : g_1(x) \leq 0, \ \dots, g_m(x) \leq 0, \quad h_1(x) = 0, \ \dots, h_p(x) = 0\}$$

con $g : R^n \to R^m, \quad g = (g_1, ..., g_m)$ e $h : R^n \to R^p, \quad (h_1, ..., h_p)$

5.3.2 Possibili Proprietà dei Domini PNL

1. Chiuso: la frontiera appartiene all'insieme
2. Limitato: rientra dentro una palla
3. Convesso: un insieme D si dice convesso se $\forall x, y \in D : \lambda x + (1-\lambda)y \in D, \forall \lambda \in [0; 1]$ oppure una condizione sufficiente affinché D sia convesso è che le g_i siano convesse e le h_j siano lineari. Quindi che l'Hessiana delle g_i sia semidefinita positiva
4. Regolare: Ne esistono diverse classi:
   (a) g_i, h_j lineari
   (b) g_i convesse, h_j lineari e sia soddisfatta la condizione di Slater (esiste un punto interno): $\exists \overline{x} : g(\overline{x}) < 0$
   (c) Soddisfa la condizione di Mangasarian: $\nabla g_i(\overline{x}), \nabla h_j(\overline{x})$ *siano linearmente indipendenti* $\quad \forall \overline{x}$ in generale vuol dire che siano linearmente indipendenti i gradienti dei vincoli attivi nei punti di intersezione. Ovvero controllare che la matrice creata con questi gradienti abbia rango massimo.

5.3.3 Sistema LKKT

Il sistema LKKT (Lagrange-Kurush-Kuhn-Tucker) è un sistema composto dalla funzione Lagrangiana e da altri vincoli basati sulla regione ammissibile, le cui soluzioni sono i punti stazionari e quindi contengono massimi e minimi del problema.

$$LKKT : \begin{cases} \nabla f(x) + \sum_{i=1}^{m} \lambda_i \cdot \nabla g_i(x) + \sum_{j=1}^{p} \mu_j \cdot \nabla h_j(x) = 0 \\ \lambda_k \cdot g_k(x) = 0 \quad \forall k \in m \\ h_l(x) = 0 \quad \forall l \in p \end{cases}$$

Tutte le soluzioni $\overline{x} = (x, \lambda, \mu)$ che risolvono il sistema sono punti stazionari.
Teorema:
Sia $\overline{x}$ in D minimo locale, allora esistono $\lambda \geq 0$ e μ in R, tali che risolvano il sistema LKKT.

Classificazione dei punti stazionari trovati:
Una $\overline{x} \in \{$*punti stazionaroi*$\}$ può essere min / max *locale*/*globale* o *Sella*.

- se λ discordi $\rightarrow$ *Sella*
- se $\lambda \geq 0 \quad \rightarrow \quad mG/mL/Sella$
- se $\lambda \leq 0 \quad \rightarrow \quad MG/ML/Sella$

Se non esiste una soluzione $\overline{x} = \{x, \lambda, \mu\}$ che soddisfa il sistema allora massimo e minimo non esistono ($\pm\infty$). Se il poliedro è limitato, una soluzione $\overline{x}$ esiste per il Teorema di Weierstrass. In un punto interno al dominio, le λ sono *zero* e le h non esistono, il sistema si riduce quindi a $\nabla f(x) = 0$.

5.3.4 Metodo di Frank-Wolfe

$$\begin{cases} \min f(x) \\ Ax \leq b \quad poliedro\ limitato \end{cases}$$

$$x^{k+1} = x^k + t_k \cdot d^k$$

Costruiamo il **Problema Linearizzato**, ovvero la restrizione ad equazione parametrica del segmento:

$$PL(x^k) \quad \begin{cases} \min \nabla f(x^k)x \\ Ax \leq b \end{cases}$$

Lo risolviamo e chiamiamo la soluzione y^k
La direzione diventa:

$$d^k = y^k - x^k$$

Il passo diventa:

$$t_k \in argmin_{t\in[0,1]}\phi$$

con $\phi = f(x^k + t \cdot d^k)$.

5.3.5 Metodo del Gradiente Proiettato

per max:

1. trovo la matrice M, ovvero la matrice formata dalle righe dei vincoli attivi di A (dominio di forma $Ax \leq b$) in x^k.
2. Calcolo la matrice H

$$H = \begin{cases} I \quad se\ x^k\ non\ ha\ vincoli\ attivi \\ I - M^T(MM^T)^{-1}M \end{cases}$$

3. calcolo la direzione

$$d^k = H \cdot \nabla f(x^k)$$

   se $d^k = 0$ vai al punto 3ª
4. calcolo il passo massimo

$$\hat{t}_k = \begin{cases} \max t \\ A(x^k + t \cdot d^k) \leq b \end{cases}$$

5. calcolo il passo effettivo

$$t_k \in argmax_{t\in[0;\hat{t}_k]}\phi(t)$$

con $\phi(t) = f(x^k + t \cdot d^k)$

6. calcolo il nuovo punto

$$x^{k+1} = x^k + t_k \cdot d^k$$

$3^{\underline{a}}$. se $d^k = 0$:

1. calcolo

$$\lambda = -(MM^T)^{-1}M \cdot \nabla f(x^k)$$

2. se $\lambda \leq 0$ allora x^k risolve $LKKT$
3. altrimenti sia $j : \lambda_j = \max_{i\in[0;k]} \lambda_i$, sappiamo che è > 0
4. elimino la riga j da M e ritorno al passo **2**.

5.3.6 Matlab Quadprog

Quadprog risolve problemi non lineari nella seguente forma:

$$\begin{cases} \min \frac{1}{2}x^T Hx + f^T x \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub \end{cases}$$

e la funzione viene invocata attraverso il seguente comando:

```
[x,fval,exitflag,output,lambda]=quadprog(h,f,a,b,aeq,beq,lb,ub,x0,options)
```

esempio: $f(x) = \frac{1}{2}x^2 + y^2 - xy - 2x - 6y$

```
h = [1 -1 ;-1 2]
f = [-2 ; -6]
a = [1 1; -1 2; 2 1]
b = [2; 2; 3]
```