UNIVERSITÀ DI PISA

Appunti Ricerca Operativa

M. Pappalardo A.A. 24/25

Christian Cantavenera | Ingegneria Informatica Unipi

A.A. 2024/25

Indice

0 Introduzione

Questo compendio rappresenta una risorsa concisa e mirata, basata sui miei appunti di Ricerca Operativa. A differenza di questi ultimi, questo documento si concentra su un numero minore di argomenti, trattati con maggiore precisione e dettaglio.

L'obiettivo è fornire una risorsa di studio più "raffinata", ideale per il ripasso degli argomenti fondamentali e la comprensione approfondita dei concetti chiave.

1 Riassunto di Programmazione Lineare (PL)

Date Importanti

- **23/09/2024:** Introduzione al corso.
- **24/09/2024:** Teoria della Programmazione Lineare (PL).
- **25/09/2024:** Problema di Assegnamento, Combinazioni convesse e coniche.
- **26/09/2024:** Teorema di Rappresentazione dei Poliedri (Weyl).
- **30/09/2024:** Teorema Fondamentale della PL, Comandi `linprog` per MATLAB.
- **01/10/2024:** Algebra della PL, Vertici del poliedro in forma duale standard, Problema di Trasporto.
- **02/10/2024:** Riepilogo degli argomenti trattati, Modelli standard.
- **03/10/2024:** Teoria della Dualità, Teorema della Dualità Forte.
- **07/10/2024:** Algoritmo del Simplesso Primale.
- **08/10/2024:** Algoritmo del Simplesso Duale.
- **09/10/2024:** Poliedro vuoto (metodo ausiliario).

1.1 1. Concetti e Definizioni Fondamentali

1.1.1 Cos'è la Ricerca Operativa?

> "La ricerca operativa si occupa di problemi di ottimizzazione con variabili continue, utilizzati per risolvere problemi decisionali."

Il corso si concentra sull'**ottimizzazione deterministica**.

Regola generale:

- Se la regione aumenta, il minimo diminuisce.
- Se la regione diminuisce, il minimo aumenta.
- Viceversa per il massimo.

1.1.2 Struttura del Corso

Il corso è diviso in 4 parti:

- **Parte 1, 2, 3:** Basate su algebra lineare.
- **Parte 4:** Basata su analisi 2.

1.1.3 Argomenti di Algebra Lineare

- Operazioni sulle matrici.
- Sistemi lineari (analisi del numero di soluzioni).
- Matrici inverse.
- Teorema del rango e soluzioni di un sistema lineare.

1.1.4 Informazioni Utili per l'Esame

- Uso di **Matlab (toolbox Optimization)**.
- È possibile (e consigliato) portare il PC all'esame.
- Struttura dell'esame:
  - **Scritto:** 4 problemi.
  - **Orale:** 4 domande.
- Non sono richieste dimostrazioni dei teoremi. È fondamentale sapere e comprenderne l'enunciato.

1.1.5 Glossario Essenziale

1. **Funzione obiettivo (FO):** Obiettivo da massimizzare o minimizzare.
2. **Vincoli:** Restrizioni sul problema.
3. **Soluzioni ammissibili:** Soluzioni che rispettano i vincoli.
4. **Regione ammissibile:** Intersezione dei semipiani (un *poliedro*).
5. **Soluzione ottima:** La soluzione ammissibile che massimizza o minimizza la FO.
6. **argmax/argmin:** Punti di massimo o minimo.
7. P**:** Indica il poliedro. **(**P**):** Indica il problema.

> *"La soluzione ottima è quella, tra le soluzioni ammissibili, che massimizza o minimizza la funzione obiettivo."*

1.1.6 Forme Standard di un Problema di PL

> **Definizione di un problema di Programmazione Lineare (PL):** Un problema di **PL** consiste nel massimo o minimo di una funzione lineare sotto vincoli lineari.

Forma Generale:

$$\begin{cases} \min / \max \ c^T x \\ Ax \leq b \\ Bx \geq d \\ Cx = e \end{cases}$$

- Variabili: $x \in \mathbb{R}^n$
- Costi/Coefficienti FO: $c \in \mathbb{R}^n$
- Matrici dei coefficienti dei vincoli: $A \in \mathbb{R}^{m_1 \times n}$, $B \in \mathbb{R}^{m_2 \times n}$, $C \in \mathbb{R}^{m_3 \times n}$
- Vettori dei termini noti: $b \in \mathbb{R}^{m_1}$, $d \in \mathbb{R}^{m_2}$, $e \in \mathbb{R}^{m_3}$

Forma Primale Standard (di massimo):

$$\begin{cases} \max \ \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \forall i = 1, \dots, m \\ x_j \geq 0 \quad \forall j = 1, \dots, n \end{cases}$$

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

- Variabili: $x \in \mathbb{R}^n$
- Costi/Coefficienti FO: $c \in \mathbb{R}^n$
- Matrice dei coefficienti dei vincoli: $A \in \mathbb{R}^{m \times n}$
- Vettore dei termini noti: $b \in \mathbb{R}^m$

  - *Problema di PL di massimizzazione di una FO lineare con vincoli lineari di $\leq$. Spesso include implicitamente $x \geq 0$ trasformando tali vincoli in $-x_j \leq 0$ e includendoli nella matrice A.*

Forma Duale Standard:

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

- Variabili: $y \in \mathbb{R}^m$ (le variabili duali)
- Costi/Coefficienti FO: $b \in \mathbb{R}^m$
- Matrice dei coefficienti dei vincoli: $A^T \in \mathbb{R}^{n \times m}$
- Vettore dei termini noti: $c \in \mathbb{R}^n$

Forma Standard (o Canonica): (Utile per l'implementazione di alcuni algoritmi, ma da non confondere con la *Forma Duale Standard*)

$$\begin{cases} \min\ c^T x \\ Ax = b \\ x \geq 0 \end{cases}$$

Formato `linprog` (MATLAB):

$$\begin{cases} \min\ C^T x \\ \mathbf{A}x \leq \mathbf{b} \\ \mathbf{A}_{eq} x = \mathbf{b}_{eq} \\ LB \leq x \leq UB \end{cases}$$

- C: vettore dei coefficienti della FO ($1 \times n$)
- $\mathbf{A}$: matrice dei coefficienti dei vincoli di disuguaglianza ($m \times n$)
- $\mathbf{b}$: vettore dei termini noti dei vincoli di disuguaglianza ($m \times 1$)
- $\mathbf{A}_{eq}$: matrice dei coefficienti dei vincoli di uguaglianza ($p \times n$)
- $\mathbf{b}_{eq}$: vettore dei termini noti dei vincoli di uguaglianza ($p \times 1$)
- LB: vettore dei limiti inferiori delle variabili ($n \times 1$)
- UB: vettore dei limiti superiori delle variabili ($n \times 1$)

1.1.7 Trasformazioni Equivalenti

Può essere trasformato seguendo i seguenti **passi**:

1. $\min f(x) \leftrightarrow \max -f(x)$**:** Il punto di ottimo non cambia ($\bar{x} \in \arg\max f \leftrightarrow \bar{x} \in \arg\min(-f)$).

2. **Vincolo** $\geq \rightarrow \leq$**:** Moltiplicare il vincolo per -1 (es. $ax \geq b \rightarrow -ax \leq -b$).

3. **Vincolo di uguaglianza $=\rightarrow\leq$:** Ogni vincolo $cx = d$ raddoppia per ogni $=$:

$$\begin{cases} cx \leq d \\ cx \geq d \quad (\text{che diventa } -cx \leq -d) \end{cases}$$

4. **Da disequazione a equazione (e viceversa):**
   - Per una disuguaglianza del tipo $a_1x_1 + \cdots + a_nx_n \leq b$, aggiungi una variabile di **slack** $S \geq 0$ per trasformarla in equazione: $a_1x_1 + \cdots + a_nx_n + S = b$. Nella funzione obiettivo $S = 0$.
   - Per una disuguaglianza del tipo $a_1x_1 + \cdots + a_nx_n \geq b$, si sottrae una variabile di **slack** $S \geq 0$: $a_1x_1 + \cdots + a_nx_n - S = b$.

5. **Variabile libera $x_i \in \mathbb{R}$:** Sostituire x_i con la differenza di due variabili non negative: $x_i = x_i^{++} - x_i^{+}$ con $x_i^{++}, x_i^{+} \geq 0$.
> Si possono mettere tutte le variabili ≥ 0.

1.1.8 Proprietà Generali di un Problema di PL

1. Il massimo, se esiste, **è sul bordo** $\rightarrow$ *poliedro illimitato* (es. primo quadrante).
2. Il problema di max può "fare" $+\infty$.
3. La soluzione ottima può non essere unica. In tal caso, tutto il segmento che congiunge due soluzioni ottime è anch'esso soluzione ottima (combinazione convessa).
4. Il poliedro può essere vuoto; ad esempio, se un'azienda presenta molti vincoli, la regione si restringe, e può capitare che i vincoli siano incompatibili $\rightarrow$ *ranking dei vincoli* $\Rightarrow$ si chiede se siano necessari o se possono essere rimossi.

1.1.9 Definizione Algebrica e Geometrica di *Poliedro*

- L'insieme delle soluzioni di un sistema lineare è dato da $\mathbf{A}x \leq \mathbf{b}$: $(x \in \mathbb{R}^n : \mathbf{A}x \leq \mathbf{b})$ con $\mathbf{A} \in \mathbb{R}^{m \times n}$ e $\mathbf{b} \in \mathbb{R}^m$.
- L'intersezione di un numero finito di semispazi chiusi ($\leq$ o $\geq$) non può contenere vincoli con $<$ o $>$.
- Se nelle istruzioni viene richiesto di risolvere < 120, scriverò ≤ 119.999. Questo perché: **La soluzione ottima non può trovarsi all'interno della regione, poiché ci si potrebbe sempre spostare in direzione di un massimo o minimo; quindi, si troverà sul bordo.**

  > – Per essere possibile in un punto interno, il gradiente si dovrebbe annullare; ma in una funzione lineare, il gradiente è $\nabla(Cx) = C$, che non è zero.

1.1.10 Soluzione Geometrica della PL in 2 Variabili

Si disegnano le rette corrispondenti ai vincoli e si determina l'intersezione dei semipiani (regione ammissibile). La direzione del gradiente $\vec{c}$ della funzione obiettivo indica la direzione di crescita. La soluzione ottima si trova sul vertice dell'ultimo vincolo toccato dalla retta nella direzione di crescita di $\vec{c}$. La soluzione ottima è un vertice solo se $\vec{c}$ NON è perpendicolare al segmento. > "È importante saper passare dalla matrice $\mathbf{A}$ e vettore $\mathbf{b}$ al grafico e viceversa."

1.1.11 Cono di Competenza

Il cono di competenza spiega come la soluzione ottima si trovi su un vertice, a meno che il gradiente non sia perpendicolare a un segmento del bordo, nel qual caso tutto il segmento è ottimo.

1.1.12 Combinazioni Convesse e Coniche

- **Combinazione Convessa:** Dati $x^1, \dots, x^k \in \mathbb{R}^n$, un punto y si dice combinazione convessa di $x^1, \dots, x^k$ se esistono $\lambda_i \in [0,1]$ per $i = 1, \dots, k$ tali che:
  1. $\sum_{i=1}^{k} \lambda_i = 1$
  2. $y = \sum_{i=1}^{k} \lambda_i x^i$

  L'insieme di tutte le possibili combinazioni convesse è l'involucro convesso ($conv(x^1, x^2, \dots, x^5)$), ed è il più piccolo insieme convesso che li contiene tutti.
- **Combinazione Conica:** Dati $x^1, \dots, x^k \in \mathbb{R}^n$, un punto $y \in \mathbb{R}^n$ si dice combinazione conica di $x^1, \dots, x^k$ se esistono $\lambda_i \geq 0$ per $i = 1, \dots, k$ tali che:

$$y = \sum_{i=1}^{k} \lambda_i x^i$$

Il cono è l'insieme di tutte le possibili combinazioni coniche.

1.2 2. Modelli di Programmazione Lineare

1.2.1 Problema di Produzione

Un'azienda massimizza il guadagno producendo beni con risorse limitate.

- **Variabili:** $x_A, x_B \in \mathbb{R}^2$ (quantità dei prodotti).
- **Funzione Obiettivo:** $\max$ (guadagno$_A \cdot x_A +$ guadagno$_B \cdot x_B$).
- **Vincoli:** Limiti sulle ore di lavoro per reparto e non negatività delle variabili.

$$\begin{cases} \text{ore reparto}_1(x_A, x_B) \leq \text{capacità}_1 \\ \dots \\ x_A, x_B \geq 0 \end{cases}$$

- Il modello rientra nella programmazione lineare poiché FO e vincoli sono lineari.
- La matrice A per un problema di produzione con m vincoli di capacità e n prodotti è di dimensione $(m+n) \times n$.

1.2.2 Problema di Assegnamento

Assegnare n lavoratori a n compiti (relazione 1:1) per minimizzare il costo totale.

- **Variabili:** $x_{ij} = 1$ se il lavoratore i svolge il compito j, 0 altrimenti. Ci sono n^2 variabili.
- **Funzione Obiettivo:** $\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$.
- **Vincoli:**
  - Ogni lavoratore svolge un solo compito: $\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i = 1, \dots, n.$ (n vincoli)
  - Ogni compito è svolto da un solo lavoratore: $\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j = 1, \dots, n.$ (n vincoli)
  - Vincoli di non negatività: $x_{ij} \geq 0$.
  - Matrice A per il problema di assegnamento con n lavoratori e n compiti è di dimensione $2n \times n^2$.

  - **Domanda da esame:** > “Se per ipotesi il valore ottimo è 130 nel caso cooperativo, in quello non cooperativo il valore aumenta o diminuisce?”

- Con l’aggiunta del vincolo $x \in \mathbb{Z}^n$ (programmazione lineare intera), la regione ammissibile si restringe e quindi il minimo della FO cresce.

1.2.3 Problema del Trasporto

Trasportare merce da m centri di produzione a n centri di domanda per minimizzare il costo totale.

- **Variabili:** x_{ij} (quantità di merce dal centro i al centro j). Ci sono $m \times n$ variabili.
- **Funzione Obiettivo:** $\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$.
- **Vincoli:**
  - L’offerta di ogni centro di produzione non può essere superata: $\sum_{j=1}^{n} x_{ij} \leq \text{offerta}_i \quad \forall i = 1, \dots, m.$ (m vincoli)
  - La domanda di ogni centro deve essere soddisfatta: $\sum_{i=1}^{m} x_{ij} \geq \text{domanda}_j \quad \forall j = 1, \dots, n.$ (n vincoli)
  - Vincoli di non negatività: $x_{ij} \geq 0$.
  - Matrice A per il problema del trasporto con m centri di produzione e n centri di domanda è di dimensione $(m + n) \times (m \times n)$.
  - Se la domanda totale è maggiore dell’offerta totale ($\sum \text{domanda}_j > \sum \text{offerta}_i$), il poliedro è vuoto.

1.3 3. Teoremi Fondamentali

1.3.1 Teorema di Rappresentazione dei Poliedri (Weyl)

Enunciato: Dato un poliedro $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, esistono un insieme finito di punti $\mathbf{V} = \{v^1, \dots, v^k\}$ (punti estremali) e un insieme finito di direzioni $\mathbf{E} = \{e^1, \dots, e^p\}$ (direzioni di recessione) tali che ogni punto $x \in P$ può essere rappresentato come una combinazione convessa dei punti di $\mathbf{V}$ più una combinazione conica delle direzioni di $\mathbf{E}$.

$$x = \sum_{i=1}^{k} \lambda_i v^i + \sum_{j=1}^{p} \mu_j e^j$$

dove $\lambda_i \geq 0$, $\sum_{i=1}^{k} \lambda_i = 1$ e $\mu_j \geq 0$. Le componenti di e^j sono dette dimensioni di recessione. Queste indicano le direzioni in cui il poliedro è illimitato.

Implicazione: Per un problema $\max c^T x$:

- Se esiste una direzione di recessione e^r tale che $c^T e^r > 0$, il problema è illimitato superiormente ($+\infty$).
- Altrimenti, il massimo si troverà su uno dei vertici v^i.

1.3.2 Teorema Fondamentale della Programmazione Lineare

Enunciato: Dato un problema $(P) : \max\{c^T x \mid Ax \leq b\}$, si assuma che:

1. Il poliedro $P = \{x \mid Ax \leq b\}$ non sia vuoto ($P \neq \emptyset$).

2. Il valore del problema sia finito ($(P) < +\infty$).

Tesi: Allora esiste almeno un vertice v^r del poliedro P che è una soluzione ottima.

$$\exists r \in \{1, \dots, k\} \text{ tale che } \max_{x \in P} c^T x = c^T v^r$$

Osservazioni:

1. Se P è limitato allora $\mathbf{E} = (\vec{0})$. Un vertice è un punto del poliedro che non si può esprimere come combinazione convessa propria di altri due punti del poliedro.

2. I poliedri limitati hanno vertici. In $\mathbb{R}^2$ i poliedri senza vertici sono: tutto il piano, semipiani, strisce, rette.

> Come so che non è infinito? Prodotto scalare $C^T e_j$, vedi lezione 26-09.
> Quanti prodotti scalari per accertarmi che abbia soluzione ce^k? Al massimo b.
> Quanti per trovare la soluzione ottima? k.

1.3.3 Teorema di Caratterizzazione dei Vertici

Enunciato: Un punto $\bar{x}$ di un poliedro è un vertice se e solo se è una soluzione di base ammissibile. > È una definizione alternativa di vertice.

1.3.4 Teorema della Dualità Forte

Ipotesi: Dati un problema primale $(P) : \max\{c^T x \mid Ax \leq b\}$ e il suo duale $(D) : \min\{y^T b \mid y^T A = c^T, y \geq 0\}$. Si assuma che entrambi i poliedri ammissibili non siano vuoti ($P, D \neq \emptyset$).

Enunciato: Allora i valori ottimi dei due problemi coincidono.

$$\max_{x \in P} c^T x = \min_{y \in D} y^T b$$

Implicazione: Se trovo una soluzione ammissibile $\bar{x}$ per il primale e una $\bar{y}$ per il duale tali che $c^T \bar{x} = \bar{y}^T b$, allora entrambe sono soluzioni ottime.
Osservazione: Una base del primale è anche una base del duale. Data una soluzione di base $\bar{x}$ per il primale, la sua soluzione duale complementare è $\bar{y}_B^T = c^T A_B^{-1}$. Se questa $\bar{y}$ è ammissibile (cioè i suoi elementi $\bar{y}_i$ con $i \in B$ sono ≥ 0), allora $\bar{x}$ è una soluzione ottima per il primale e la soluzione duale $\bar{y}$ (con $\bar{y}_N = 0$) è ottima per il duale.

1.4 4. Algebra della PL e Algoritmi

1.4.1 Calcolo dei Vertici (Forma Primale Standard)

1. Un **vertice** è una **soluzione di base ammissibile**.

2. Una **base** è un insieme B di n indici di riga (vincoli) tale che la sottomatrice A_B (ottenuta estraendo le righe A_i con $i \in B$) sia invertibile ($det(A_B) \neq 0$). A_B è di dimensione $n \times n$.

3. La **soluzione di base** associata a B è $\bar{x} = A_B^{-1} b_B$.

4. La soluzione di base è **ammissibile** (e quindi è un vertice) se soddisfa tutti gli altri vincoli: $A_N\bar{x} \leq b_N$.

5. Una soluzione di base è **degenere** se soddisfa con l'uguaglianza più di n vincoli.
    - Due soluzioni di base sono degeneri se due basi generano lo stesso vertice.

> - Degenere significa sovrabbondante? **No**, in $\mathbb{R}^2$ è più facile pensarlo ma in dimensioni $n > 2$ questo non si verifica.
>
> - Si può vedere se un vincolo è eliminabile? Non c'è un algoritmo.
>
> - n iperpiani in $\mathbb{R}^n$ danno una soluzione.

1.4.2 Vertici del Poliedro in Forma Duale Standard

Per un problema in forma duale standard $P = \{x \mid Ax = b, x \geq 0\}$:

- Si dividono le **variabili!** in due blocchi: di base (x_B) e non di base (x_N). Le variabili non di base si pongono a 0.
- La soluzione di base è $x_B = A_B^{-1}b$.
- Un punto x di un poliedro duale standard è un vertice se e solo se è una soluzione di base ammissibile (cioè $x_B \geq 0$).
- Una soluzione è degenere se uno degli elementi di x_B è 0.
- **Domanda da esame:** > "Quanti sono i vertici di un poliedro?" Il numero massimo di vertici è dato dal coefficiente binomiale $\binom{m}{n} = \frac{m!}{n!(m-n)!}$, dove m è il numero di vincoli e n è il numero di variabili.
- **Domanda da esame:** > Quante variabili e vincoli ci sono in un problema di assegnamento 50×50? 2500 variabili e 2700 vincoli (*100 per ogni assegnamento, i vincoli si raddoppiano, quindi 200*).

1.4.3 Algoritmo del Simplesso Primale

Serve a risolvere problemi del tipo $\max\{c^Tx \mid Ax \leq b\}$. Si parte da un vertice ammissibile e ci si sposta su vertici adiacenti migliorando la funzione obiettivo.

Passi:

1. **Inizializzazione:** Parti da una base ammissibile B (un vertice $\bar{x}$).

2. **Test di ottimalità:** Calcola la soluzione duale complementare $\bar{y}_B = c^T {A_B^{-1}}^T$.
    - Se $\bar{y}_B \geq 0$, la soluzione corrente $\bar{x}$ è **ottima**. **STOP**.
    - Altrimenti, non è ottima.

3. **Scelta dell'indice uscente (h):** Scegli un indice $h \in B$ tale che $\bar{y}_h < 0$. Per la regola di Bland, scegli il minimo indice h. L'indice h esce dalla base.

4. **Calcolo della direzione:** Calcola la direzione di spostamento $W^h = -(A_B^{-1})_{\cdot h}$ (la colonna h-esima di $-A_B^{-1}$). Le semirette uscenti da $\bar{x}$ sono del tipo $\bar{x} + \lambda W^i (\lambda \geq 0)$.

5. **Test di illimitatezza:** Se tutti i prodotti scalari $A_i W^h \leq 0$ per ogni $i \notin B$, il problema è **illimitato superiormente**. **STOP**.

6. **Scelta dell'indice entrante (k):** Calcola i rapporti $r_i = \frac{b_i - A_i\bar{x}}{A_i W^h}$ per tutti gli indici $i \notin B$ tali che $A_i W^h > 0$. Scegli l'indice k corrispondente al rapporto minimo. L'indice k entra nella base.

   > • All'esame devo:
   >
   > – 1. Calcolare il minimo dei rapporti $r_i = \frac{b_i - A_i\bar{x}}{A_i W^h}$ con $i \in N \quad A_i W^h > 0$.
   > – 2. Faccio i prodotti scalari al denominatore e prendo solo quelli positivi (nel nostro caso sono 4 prodotti, quelli dei restanti indici non adiacenti).
   > – 3. Faccio i rapporti (al massimo 4).

7. **Aggiornamento:** La nuova base è $B_{new} = (B \quad \{h\}) \cup \{k\}$. Torna al passo 2.

   > • L'algoritmo termina in un numero finito di passi: non si ritorna su un vertice già visitato.

1.4.4 Algoritmo del Simplesso Duale

Si applica quando si ha una base **ammissibile per il duale ma non per il primale** ($\bar{y} \geq 0$ ma $\bar{x}$ non ammissibile).

Passi:

1. **Inizializzazione:** Parti da una base B tale che $\bar{y}_B = c^T {A_B^{-1}}^T \geq 0$ (ammissibilità duale).

2. **Test di ottimalità:** Verifica l'ammissibilità primale calcolando $\bar{x} = A_B^{-1} b_B$.

   > • Se $A_N \bar{x} \leq b_N$, la soluzione $\bar{x}$ è ammissibile e quindi **ottima**. **STOP**.
   > • Altrimenti, non è ottima.

3. **Scelta dell'indice entrante (k):** Scegli un indice $k \notin B$ che corrisponde a un vincolo violato: $A_k \bar{x} > b_k$. Per la regola di Bland, scegli il minimo indice k. L'indice k entra nella base.

4. **Calcolo della direzione:** Calcola la riga A_k e le colonne di $W = -A_B^{-1}$.

5. **Test di inammissibilità:** Se tutti i prodotti scalari $A_k W^i \geq 0$ per ogni $i \in B$, il problema primale è **vuoto**. **STOP**.

6. **Scelta dell'indice uscente (h):** Calcola i rapporti $r_i = \frac{-\bar{y}_i}{A_k W^i}$ per tutti gli indici $i \in B$ tali che $A_k W^i < 0$. Scegli l'indice h corrispondente al rapporto minimo. L'indice h esce dalla base.

   > • quali sono i rapporti del simplesso primale? $\frac{b_i - A_i\bar{x}}{A_i W^h}$ con $i \in N$ e $A_i W^h > 0$.
   >
   > • cosa indicano i rapporti r? circa la lunghezza dello spostamento lungo lo spigolo scelto prima di uscire dal poliedro.
   >
   > • come trovo le direzioni degli spigoli adiacenti? $\bar{x} + \lambda W^h$.
   >
   > • r può fare quindi zero? mi turberebbe parecchio, ma sì (nei casi degeneri).

7. **Aggiornamento:** La nuova base è $B_{new} = (B \quad \{h\}) \cup \{k\}$. Torna al passo 2.

   > • Domanda, quanto vale la FO nel duale per $\bar{y} = (0, 0, 0, -\frac{16}{7}, -\frac{1}{7}, 0)$? -37.
   >
   > • Domanda difficile: quanto è diminuito 17? (ora è $\frac{11}{3}$).

1.4.5 Verificare se un Poliedro è Vuoto (Metodo Ausiliario)

Per un poliedro in forma duale standard $P = \{x \mid Ax = b, x \geq 0\}$, si può usare un problema ausiliario per trovare un vertice di partenza o dimostrare che è vuoto.

Problema Duale Ausiliario (DA):

$$(DA) \begin{cases} \min \sum \epsilon_i \\ Ax + I\epsilon = b \\ x \geq 0, \epsilon \geq 0 \end{cases}$$

- Si parte dalla soluzione di base ammissibile ovvia $x = 0, \epsilon = b$ (assumendo $b \geq 0$; se $b_i < 0$ per qualche i, moltiplicare la riga per -1).
- Si risolve il (DA) con il simplesso.
- **Teorema:**
  - Se il valore ottimo del (DA) è **maggiore di zero**, il poliedro originale P è **vuoto**.
  - Se il valore ottimo del (DA) è **zero**, il poliedro P **non è vuoto** e la soluzione ottima del (DA) fornisce una base ammissibile per iniziare a risolvere il problema originale.

1.4.6 Preambolo Programmazione Lineare Intera (PLI)

Quando le variabili devono assumere valori interi ($x \in \mathbb{Z}^n_+$), la regione ammissibile si restringe rispetto al caso di variabili continue.

- Il valore ottimo della PLI (v_{PLI}) è sempre minore o uguale al valore ottimo della PL continua (v_{PL}), ovvero $v_{PLI} \leq v_{PL}$.
- Una soluzione ottima di un problema PLI non è necessariamente un vertice del poliedro continuo.

> Nessun produttore, ad esempio, di capi vorrà produrre 43.57 pantaloni e 22.69 gonne.

2 Riassunto di Programmazione Lineare Intera (PLI)

Date Importanti

- **10/10/2024:** Problema di Caricamento (Knapsack), Teorema di Caricamento.
- **14/10/2024:** Esempi su Zaino Binario e Intero, riassunto regole di Bland.
- **15/10/2024:** Problema di Trasporto e Assegnamento Intero, Teorema di Unimodularità.
- **16/10/2024:** Problema di Bin Packing, algoritmi NFD, FFD, BFD.
- **17/10/2024:** Piani di Taglio e Algoritmo di Riduzione del Gap (Gomory).
- **21/10/2024:** Esercizi sui Piani di Taglio.
- **22/10/2024:** Problema del Commesso Viaggiatore (TSP).
- **23/10/2024:** Tecniche euristiche per lo zaino e rilassamenti, TSP simmetrico e 1-albero.
- **24/10/2024:** Modello TSP Simmetrico e domande specifiche.
- **29/10/2024:** Metodo del Branch and Bound.
- **30/10/2024:** Problema di Copertura (Covering) e sue varianti.

2.1 1. Introduzione alla PLI

La Programmazione Lineare Intera (PLI) è una branca dell'ottimizzazione che si occupa di problemi in cui le variabili decisionali sono vincolate ad assumere valori interi. Questo vincolo aggiuntivo, rispetto alla Programmazione Lineare (PL), modella in modo più realistico scenari in cui non sono ammesse soluzioni frazionarie (es. produrre mezzo aereo, assegnare un lavoratore a 0.7 compiti).

Relazione tra PL e PLI: Un qualsiasi problema di PLI può essere "rilassato" rimuovendo il vincolo di interezza, ottenendo un problema di PL.

- La regione ammissibile di un problema PLI, indicata con $S = \{x \in \mathbf{Z}^n_+ : Ax \leq b\}$, è un sottoinsieme discreto (i punti a coordinate intere) della regione ammissibile del suo rilassamento continuo, $P = \{x \in \mathbf{R}^n_+ : Ax \leq b\}$.
- Il valore ottimo della PLI (v_{PLI}) è sempre peggiore o uguale a quello del suo rilassamento continuo (v_{PL}).
  - Per problemi di **massimo**: $v_{PLI} \leq v_{PL}$.
  - Per problemi di **minimo**: $v_{PLI} \geq v_{PL}$.
- La soluzione ottima del rilassamento continuo, se arrotondata, non garantisce né l'ammissibilità né l'ottimalità per il problema PLI.

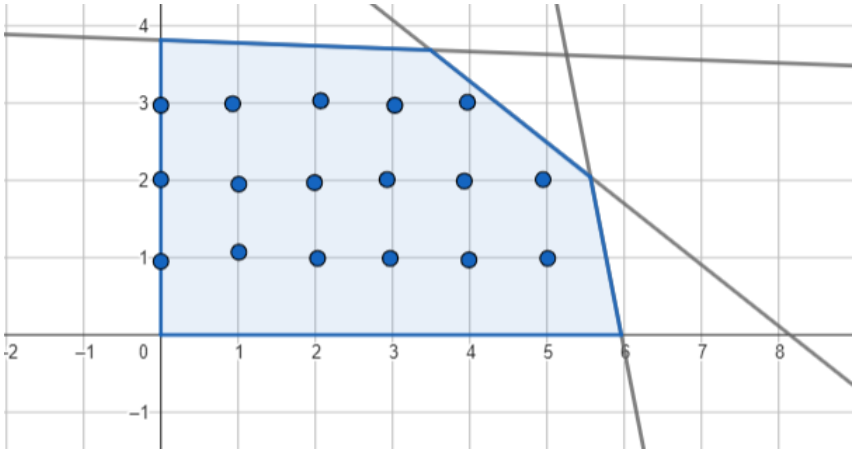

Figura 2.1: Confronto tra PL e PLI

L'insieme dei punti ammissibili della PLI è S, mentre il poliedro del rilassato continuo è P. L'obiettivo della PLI è spesso quello di trovare un modo efficiente per caratterizzare l'involucro convesso dei punti ammissibili, $conv(S)$, poiché massimizzare su S è equivalente a massimizzare su $conv(S)$.

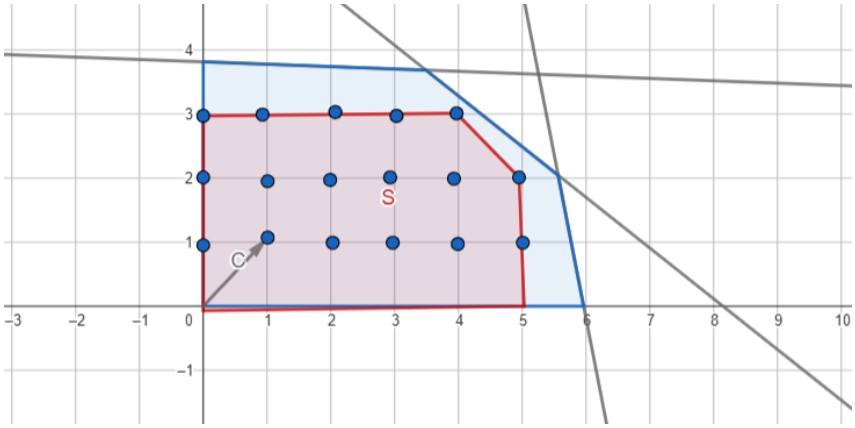

Figura 2.2: Inviluppo Convesso

$$\begin{cases} \max\ c^T x \\ x \in S \end{cases} \equiv \begin{cases} \max\ c^T x \\ x \in conv(S) \end{cases} \leq \begin{cases} \max\ c^T x \\ x \in P \end{cases}$$

2.2 2. Problema di Caricamento (Knapsack Problem)

> Anche chiamato *knapsack problem*, il **problema dello zaino**.

È un problema classico di PLI in cui si devono scegliere quali oggetti inserire in uno zaino con capacità limitata per massimizzare il valore totale.

2.2.1 Formulazione del Problema

Esistono due formulazioni principali:

- **Zaino Booleano (0/1 Knapsack):** Ogni oggetto può essere preso una sola volta.
- **Zaino Intero (Unbounded Knapsack):** Ogni oggetto può essere preso più volte.

Zaino Booleano

$$\begin{cases} \max\ v^T x \\ p^T x \leq P \\ x \in \{0,1\}^n \end{cases}$$

Zaino Intero

$$\begin{cases} \max\ v^T x \\ p^T x \leq P \\ x \in \mathbb{Z}_+^n \end{cases}$$

dove:

- v è il vettore dei valori degli oggetti.
- p è il vettore dei pesi (ingombri) degli oggetti.
- P è la capacità massima dello zaino.
- x è il vettore delle variabili decisionali.

> In MATLAB, questi problemi si risolvono con `intlinprog (c, I, A, b, Aeq, beq, LB, UP)`, dove `I` specifica gli indici delle variabili intere.

2.2.2 Limiti Inferiori (v_i) e Superiori (v_s)

Per stimare il valore ottimo v_{PLI} senza risolvere il problema complesso, si usano limiti inferiori e superiori rapidi da calcolare.

- v_i **(Lower Bound):** È sempre il valore di una **soluzione ammissibile**, spesso trovata con un algoritmo *greedy* (rapido).
- v_s **(Upper Bound):** È una stima per eccesso (per problemi di max) o per difetto (per problemi di min), spesso ottenuta dal **rilassamento continuo** del problema.

Riepilogo delle Regole:

- Per un problema di **massimo** (come lo Zaino):

$$\text{sol. amm. (greedy) } v_i \leq v_{PLI} \leq v_s = \lfloor v_{ril.cont.} \rfloor \text{ (valore ottimo del rilassato continuo)}$$

- Per un problema di **minimo**:

$$\text{ril. cont. } v_i = \lceil v_{ril.cont.} \rceil \leq v_{PLI} \leq v_s \text{ (sol. amm. greedy)}$$

> **Domanda da esame:** "Se vi chiedo l'errore: $\frac{v_s - v_i}{v_i}$" **Domanda Difficile:** Ho un problema di massimo; trovo l'ottimo del rilassato continuo, se arrotondo per difetto la *soluzione* è sempre ammissibile? **No, potrei violare il vincolo di capacità**.

2.2.3 Teorema: Soluzione Ottima del Rilassamento Continuo

Il concetto chiave è il **rendimento** ($r_i = v_i/p_i$), che misura il rapporto valore/peso di ogni oggetto.

- **Zaino Intero:** La soluzione ottima del rilassato continuo si ottiene saturando completamente lo zaino con l'oggetto a **massimo rendimento**.

- **Zaino Booleano:** La soluzione ottima del rilassato continuo si ottiene caricando gli oggetti in ordine di rendimento decrescente. Quando un oggetto non entra interamente, se ne carica una frazione per saturare esattamente lo zaino.

> **Domanda da esame:** Conviene caricare la stiva col bene più leggero? *Dipende*, il vero valore importante è il **rendimento**.

2.2.4 Esempio (Zaino Booleano e Intero)

Dati: Valori $v = [10, 17, 22, 21]$, Pesi $p = [4, 5, 6, 2]$, Capacità $P = 7$. Rendimenti: $r = (2.5, 3.4, 3.66, 10.5)$. L'ordine di rendimento è x_4, x_3, x_2, x_1.

Zaino Booleano:

- v_i **(soluzione ammissibile greedy):**

  1. Prendo l'oggetto a massimo rendimento: $x_4 = 1$ (peso 2, valore 21). Spazio residuo: 5.
  2. Provo il secondo: $x_3 = 1$ (peso 6). Non entra.
  3. Provo il terzo: $x_2 = 1$ (peso 5). Entra. Spazio residuo: 0.

  - Soluzione ammissibile: $x = (0, 1, 0, 1)$, $v_i = 17 + 21 = 38$.

- v_s **(rilassamento continuo):**

  1. Prendo $x_4 = 1$. Spazio residuo: 5.
  2. L'oggetto successivo, x_3, non entra interamente. Ne prendo una frazione: $x_3 = 5/6$.

  - Soluzione del rilassato: $x_{rc} = (0, 0, 5/6, 1)$.
  - Valore del rilassato: $v_{rc} = (5/6) \cdot 22 + 1 \cdot 21 \approx 39.33$.
  - $v_s = \lfloor 39.33 \rfloor = 39$.

- **Gap:** $38 \leq v_{PLI} \leq 39$. Errore: $1/38$.

Zaino Intero:

- v_i **(soluzione ammissibile greedy):**

  - Prendo solo l'oggetto a massimo rendimento, x_4, quante più volte possibile: $\lfloor 7/2 \rfloor = 3$ volte.
  - Soluzione ammissibile: $x = (0, 0, 0, 3)$, $v_i = 3 \cdot 21 = 63$.

- v_s **(rilassamento continuo):**

  - Saturo lo zaino con l'oggetto a massimo rendimento: $x_4 = 7/2$.
  - Soluzione del rilassato: $x_{rc} = (0, 0, 0, 7/2)$.
  - Valore del rilassato: $v_{rc} = (7/2) \cdot 21 = 73.5$.
  - $v_s = \lfloor 73.5 \rfloor = 73$.
  - Negli esercizi basta usa linprog per trovare $\bar{x}$

- **Gap:** $63 \leq v_{PLI} \leq 73$. Errore: $10/63$.

2.3 3. Problemi di Trasporto, Assegnamento e Unimodularità

2.3.1 Teorema (6): Equivalenza tra PL e PLI

Un problema di PLI è equivalente al suo rilassamento continuo PL se e solo se tutti i vertici del poliedro del rilassato continuo P hanno coordinate intere. In questo caso, la soluzione ottima del problema PL, trovata con algoritmi come il Simplesso, sarà automaticamente intera e quindi ottima anche per il problema PLI.

$$\text{Se } \bar{x}_{rc} \in \mathbb{Z}^n \implies v_{PLI} = c^T \bar{x}_{rc}$$

> Si può definire un teorema **astratto** di equivalenza, perché non è risolutivo, non ci dice come trovare $conv(S)$.

2.3.2 Teorema: Unimodularità della Matrice di Trasporto

La matrice dei vincoli A di un **problema di trasporto** (e del **problema di assegnamento**, che ne è un caso particolare) è **totalmente unimodulare**. Questo significa che ogni sua sottomatrice quadrata invertibile ha determinante $+1$ o -1.

Implicazione Fondamentale: A causa di questa proprietà, i vertici del poliedro di un problema di trasporto sono **sempre a coordinate intere**.

> **Conclusione:** Per risolvere problemi di trasporto e assegnamento, si può usare `linprog` (per PL) invece di `intlinprog` (per PLI), poiché la soluzione del rilassato continuo sarà garantita intera.

2.4 4. Problema di Bin Packing

L'obiettivo è minimizzare il numero di contenitori ("bin") di capacità fissa P necessari per contenere un insieme di oggetti di pesi diversi p_j.

2.4.1 Formulazione del Modello

- **Variabili:**
  - $x_{ij} = 1$ se l'oggetto j è nel contenitore i, 0 altrimenti.
  - $y_i = 1$ se il contenitore i è usato, 0 altrimenti.
- **Modello:**

$$\begin{cases} \min \sum_{i=1}^{m} y_i \\ \sum_{i=1}^{m} x_{ij} = 1 \quad \forall j & \text{(ogni oggetto è in un solo contenitore)} \\ \sum_{j=1}^{n} p_j x_{ij} \leq P \cdot y_i \quad \forall i & \text{(la capacità di ogni bin non è superata)} \\ x_{ij}, y_i \in \{0,1\} \end{cases}$$

> Il vincolo di capacità è cruciale: se un contenitore non è usato ($y_i = 0$), il lato destro della disuguaglianza diventa 0, forzando tutti gli x_{ij} per quel contenitore ad essere 0.

2.4.2 Limiti Inferiori (v_i) e Superiori (v_s)

Siamo in un problema di minimo.

- v_i **(Lower Bound):** È il numero minimo teorico di bin, dato dalla somma totale dei pesi divisa per la capacità di un bin, arrotondata per eccesso.
$$v_i = \left\lceil \frac{\sum_{j=1}^{n} p_j}{P} \right\rceil$$
- v_s **(Upper Bound):** È una soluzione ammissibile, trovata con algoritmi euristici.

2.4.3 Algoritmi Euristici per v_s

Questi algoritmi ordinano gli oggetti per peso decrescente e poi li assegnano ai bin.

1. **Next-Fit-Decreasing (NFD):** Si prova a inserire l'oggetto corrente nel bin *attualmente aperto*. Se non entra, si chiude il bin attuale e se ne apre uno nuovo per l'oggetto. È veloce ma poco efficiente.
    1. Apro il primo contenitore
    2. Prendo il primo oggetto, ci entra?
        1. Si , next obj
        2. No, *poliedro vuoto!*
    3. Prendo il secondo oggetto, ci entra?
        1. Si, next obj
        2. No, next container
    4. ….
2. **First-Fit-Decreasing (FFD):** Per ogni oggetto, si scorrono i bin *dall'inizio* e lo si inserisce nel primo in cui c'è spazio sufficiente. Più lento ma più efficiente di NFD.
3. **Best-Fit-Decreasing (BFD):** Per ogni oggetto, si controllano *tutti* i bin aperti e lo si inserisce in quello che lascerebbe lo **sfrido** (spazio residuo) minore. È il più lento ma generalmente il più efficace. *Proviamo il BFD su* $\boxed{865443}$

Cont	Peso	Sfrido
1	$8^{(1)}$	2
2	$6^{(2)}, 4^{(4)}$	~~4~~0
3	$5^{(3)}, 4^{(4)}$	~~5~~1
4	$3^{(6)}$	7

2.4.4 Comandi linprog

variabili: $x_{11}, x_{12}, ..., x_{16}, x_{21}, ..., x_{46}, y_1, y_2, y_3, y_4$ $(28 \text{ tot.} = 24x + 4y)$

$$A = \begin{bmatrix} 4 \times 28 \end{bmatrix} \quad b = \underset{10y \text{ a sx}}{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}} \quad Aeq = \begin{bmatrix} 6 \times 28 \end{bmatrix} \quad beq = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad LB = \overset{28\times1}{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}} \quad UB = \overset{28\times1}{\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}}$$

Nel formato primale del rilassato continuo $x_{ij} \in \{0, 1\}$ e $y_i \in \{0, 1\}$ li avremmo scritti

$$24 \begin{cases} -x \leq 0 \\ x \geq 1 \end{cases} \quad 0 \leq x \leq 1 \qquad 4 \begin{cases} -y \leq 0 \\ y \geq 1 \end{cases} \quad 0 \leq y \leq 1$$

dunque le righe della matrice A sarebbero 72, $(6 \times 2 + 4 + 2 \times 24 + 2 \times 4)$

2.5 5. Piani di Taglio (Algoritmo di Riduzione del Gap)

L'idea è di risolvere il rilassamento continuo P e, se la soluzione ottima x_{rc} non è intera, aggiungere un nuovo vincolo lineare (un **piano di taglio**) che "tagli via" x_{rc} senza eliminare alcuna soluzione intera ammissibile $x \in S$.

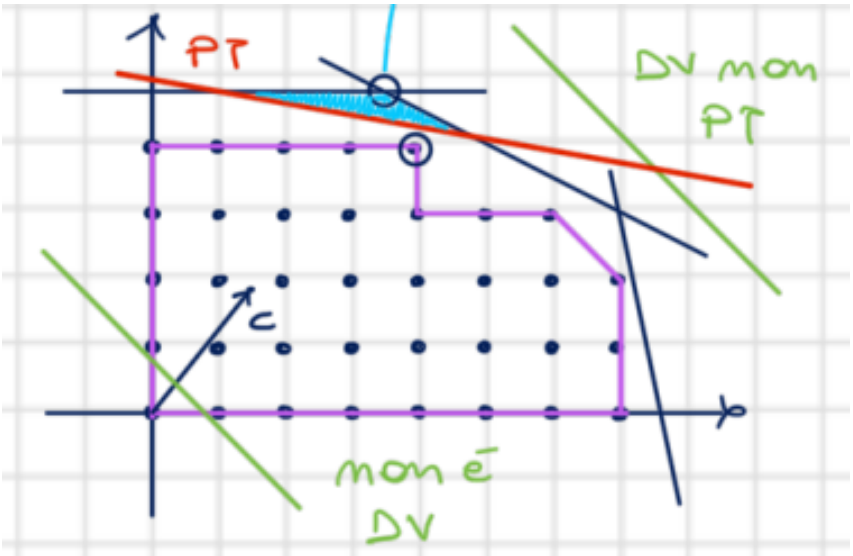

Figura 2.3: Piani di Taglio

Definizioni:

- **Disuguaglianza Valida:** Una disuguaglianza $\gamma^T x \leq \gamma_0$ soddisfatta da tutti i punti $x \in S$.
- **Piano di Taglio:** Una disuguaglianza valida che è violata dalla soluzione ottima del rilassato continuo corrente, x_{rc} (cioè $\gamma^T x_{rc} > \gamma_0$).

Aggiungendo piani di taglio si restringe il poliedro, avvicinandolo a $conv(S)$, e si riduce il gap tra v_{PLI} e v_{PL}.

$$V_{PLI} \leq V_{P_{nuovo}} < V_{PL}$$

2.5.1 Teorema 8: Taglio di Gomory

Questo teorema fornisce una formula per generare un piano di taglio a partire da una soluzione di base non intera. Sia il problema in forma duale standard ($Ax = b, x \geq 0$) e sia $\bar{x}_{rc}$ la soluzione di base ottima. Sia r una riga della base tale che la componente x_{rc_r} sia non intera.

Il **piano di taglio di Gomory** è:

$$\sum_{j \in N} \{\tilde{a}_{rj}\} x_j \geq \{x_{rc_r}\}$$

dove:

- N è l'insieme degli indici delle variabili non di base.
- $\tilde{A} = A_B^{-1} A_N$ è la parte del tableau del simplesso relativa alle variabili non di base.
- $\{a\} = a - \lfloor a \rfloor$ è la parte frazionaria di a (es. $\{3.14\} = 0.14$, $\{-3.14\} = -3.14 - (-4) = 0.86$).

2.5.2 Algoritmo di Riduzione del Gap (Passi)

1. Risolvi il rilassato continuo e trova la soluzione ottima x_{rc}. Se è intera, STOP.
2. Porta il problema in forma duale standard ($Ax = b, x \geq 0$) aggiungendo variabili di slack etc etc.
3. Identifica la base B che ha generato x_{rc} e scegli una riga r dove x_{rc_r} non è intero.
4. Calcola $\tilde{A} = A_B^{-1} A_N$.
5. Scrivi il piano di taglio di Gomory usando la riga r di $\tilde{A}$ e il valore x_{rc_r}.
6. Aggiungi il nuovo vincolo al problema e torna al passo 1.
7. Per convertire da variabili slack a variabili originali, sostituisre il lato sinistro della disequazione finale con $x^-_{rc_r} - x_r$

> **Domanda da esame:** L'ottimo del rilassato continuo verifica il piano di taglio? **No!** Il piano di taglio è progettato per tagliare via l'ottimo del rilassato continuo corrente.

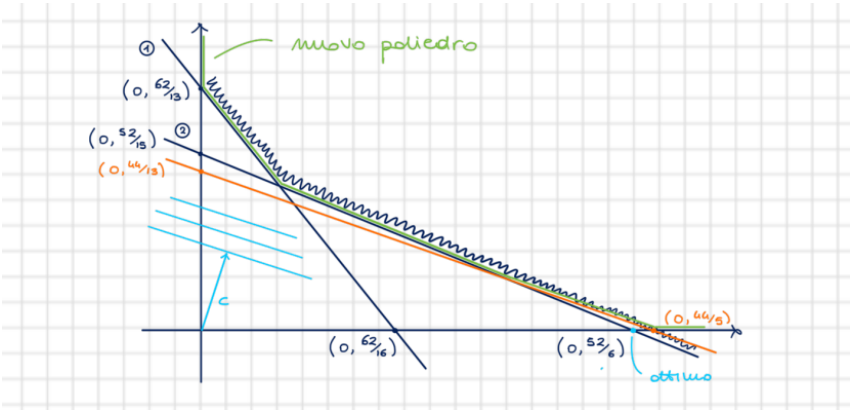

Figura 2.4: Esempio Poliedro con Taglio

2.6 6. Problema del Commesso Viaggiatore (TSP)

L'obiettivo è trovare il percorso di costo minimo che visiti un insieme di città una sola volta e ritorni alla città di partenza (ciclo Hamiltoniano).

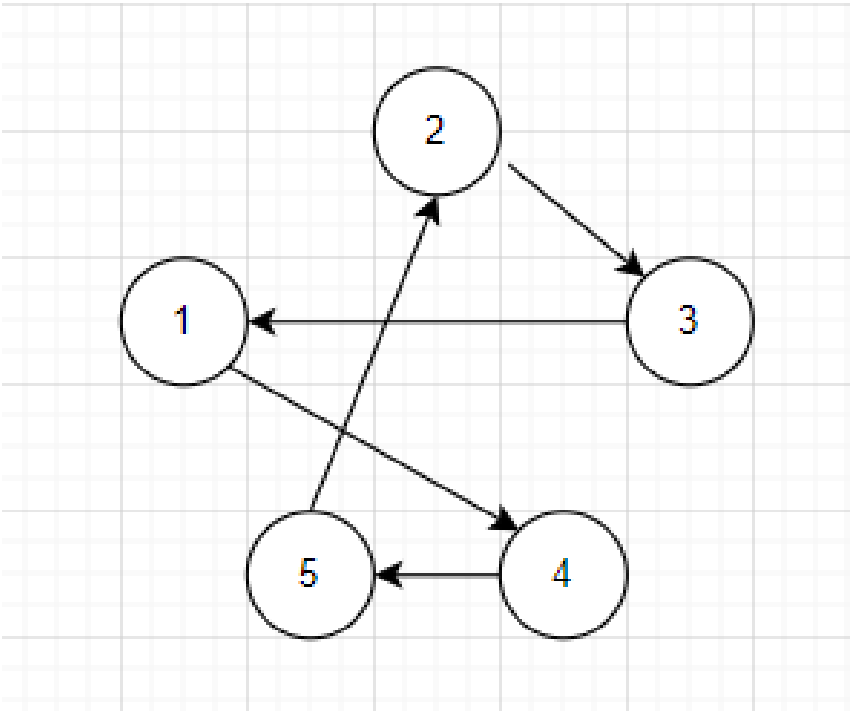

Figura 2.5: Ciclo Hamiltoniano

2.6.1 TSP Asimmetrico ($C_{ij} \neq C_{ji}$)

Formulazione: Il problema può essere modellato come un **problema di assegnamento** con l'aggiunta di vincoli per eliminare i sotto-cicli (tour che non includono tutte le città).

- **Modello di Assegnamento (incompleto):**

$$\begin{cases} \min \sum_{i,j} c_{ij} x_{ij} \\ \sum_j x_{ij} = 1 \quad \forall i \quad \text{(da ogni città esce un arco)} \\ \sum_i x_{ij} = 1 \quad \forall j \quad \text{(in ogni città entra un arco)} \\ x_{ij} \in \{0,1\} \end{cases}$$

  > Questo modello da solo non basta, perché permette soluzioni come due cicli disgiunti (es. 1-2-1 e 3-4-5-3).

- **Vincoli di Eliminazione dei Sotto-cicli:** Per ogni sottoinsieme proprio non vuoto di città $S \subset N$, si deve imporre che almeno un arco esca da S per andare in una città non in S.

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

  > Il numero di questi vincoli è esponenziale ($2^n - 2$), rendendo il modello difficile da risolvere direttamente. Il numero di connessioni in un grafo da n nodi è $2^n - 2n - 2$

$|S| = 3 \rightarrow 123, 124, 125, 234, 235, 245, 345, 134, 135, 145$
Preso $S = 234 \rightarrow x_{21} + x_{25} + x_{31} + x_{35} + x_{41} + x_{45} \geq 1$

Limiti v_i e v_s per TSP Asimmetrico:

- v_i **(Lower Bound):** Si ottiene risolvendo il rilassamento del problema, cioè il **problema di assegnamento di costo minimo** (ignorando i vincoli di sotto-ciclo).
- v_s **(Upper Bound):** Si parte dalla soluzione dell'assegnamento. Se questa contiene sotto-cicli, si applica l'**Algoritmo di fusione dei cicli (delle toppe)** per unirli in un unico ciclo Hamiltoniano. Si scelgono gli archi da scambiare per minimizzare l'aumento di costo.

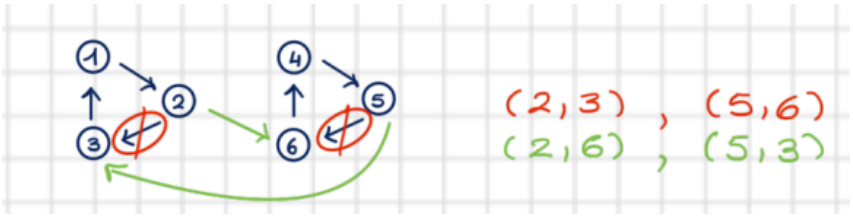

Figura 2.6: Fusione Cicli

Supponiamo due cicli Hamiltoniani (1,2,3) e (4,5,6) disgiunti

- Selezioniamo un arco del primo ciclo (i, j) (es. 23) e uno del secondo (k, l) (es,56)
- Eliminiamoli
- Incrociamoli inserendo altri due archi $(i, l)|(k, j)$ (es: 26 e 53 *notare l'incrocio*)
- Abbiamo un ciclo

La nostra $v_s = v_{ass} - c_{23} - c_{56} + c_{26} + c_{53}$, si prende la fusione più conveniente. Sempre > 0, se non lo fosse, il nuovo assegnamento sarebbe inferiore a v_i

---

2.6.1.1 Tecniche euristiche *greedy*

2.6.1.2 Per trovare *Soluzione Ammissibile*

- **Metodo 1**: esaminiamo gli oggetti in ordine di **valore decrescente**;
  ogni oggetto viene inserito purché sia rispettato il vincolo di capacità.
  es.

oggetto	1	2	3	4	5	6	7	8	9
valore	50	65	35	16	18	55	45	40	25
peso	31	39	26	21	25	28	29	27	23

$C = 100$

$$x_2 = 1,\ x_6 = 1,\ x_1 = 1,\ x_7 = 0,\ x_8 = 0,\ x_3 = 0,\ x_9 = 0,\ x_5 = 0,\ x_4 = 0$$

$$\Rightarrow V_1(P) = 170$$

- **Metodo 2**: esaminiamo gli oggetti in ordine di **peso crescente**;
  ogni oggetto viene inserito purché sia rispettato il vincolo di capacità.
  es.

$$x_4 = 1,\ x_9 = 1,\ x_5 = 1,\ x_3 = 1,\ x_8 = 0,\ x_6 = 0,\ x_7 = 0,\ x_1 = 0,\ x_2 = 0$$

$$\Rightarrow V_1(P) = 94$$

- **Metodo 3**: esaminiamo gli oggetti in ordine di **rendimento** (valore/peso) **decrescente**;
  ogni oggetto viene inserito purché sia rispettato il vincolo di capacità.
  es.

$$x_6 = 1,\ x_2 = 1,\ x_1 = 1,\ x_7 = 0,\ x_8 = 0,\ x_3 = 0,\ x_9 = 0,\ x_4 = 0,\ x_5 = 0$$

$$\Rightarrow V_1(P) = 170$$

2.6.1.3 Rilassamenti

Supponiamo che le variabili siano in ordine di **rendimento decrescente**;
sia h l'indice tale che:

$$\sum_{j=1}^{h} p_j \leq C \quad \text{e} \quad \sum_{j=1}^{h+1} p_j > C$$

Il **rilassamento continuo**:

$$\begin{cases} \max \sum_{j=1}^{m} v_j x_j \\ \sum_{j=1}^{m} p_j x_j \leq C \\ 0 \leq x_j \leq 1 \end{cases}$$

ha come soluzione ottima:

$$\bar{x}_1 = 1, \dots, \bar{x}_h = 1, \quad \bar{x}_{h+1} = \frac{C - \sum_{j=1}^{h} p_j}{p_{h+1}}, \quad \bar{x}_{h+2} = 0, \dots, \bar{x}_n = 0$$

2.6.2 TSP Simmetrico ($C_{ij} = C_{ji}$)

Si usano meno variabili (x_{ij} con $i < j$).
12 13 14 15 23 24 25 34 35 45 $\rightarrow \frac{n^2-n}{2}$ variabili, la metà

Limiti v_i e v_s per TSP Simmetrico:

- v_s **(Upper Bound):** Trovata con algoritmi euristici come **l'Algoritmo del Nodo Più Vicino (Nearest Neighbor)**. Si parte da un nodo e ci si sposta iterativamente al nodo non ancora visitato più vicino, finché tutti sono stati visitati, per poi tornare al punto di partenza. Non garantisce l'ottimalità.
- v_i **(Lower Bound):** Si usa il rilassamento basato sul **k-Albero di costo minimo**.
  - Un **1-Albero** è un albero di copertura sui nodi $\{2, \dots, n\}$ più i due archi di costo minimo che collegano il nodo 1 al resto dell'albero.
  - **Algoritmo per v_i:**
    1. Isola il nodo 1.
    2. Trova l'**Albero di Copertura di Costo Minimo (MST)** sui nodi $\{2, \dots, n\}$ usando l'**Algoritmo di Kruskal**.
       1. Ordino tutti gli archi del grafo in ordine crescente di costo (peso).
       2. Parto con un insieme vuoto di archi.
       3. Aggiungo l'arco di costo minore che non forma un ciclo con gli archi già aggiunti.
       4. Ripeto il passo 3 fino a quando non ho aggiunto n - 1 archi (dove n è il numero di vertici).
       5. Al termine, l'insieme di archi scelti costituisce l'albero di copertura minimo.
    3. Aggiungi i due archi di costo minimo che collegano il nodo 1 ai nodi dell'MST.
       - Il costo totale di questo 1-Albero è una valida valutazione inferiore per il TSP.

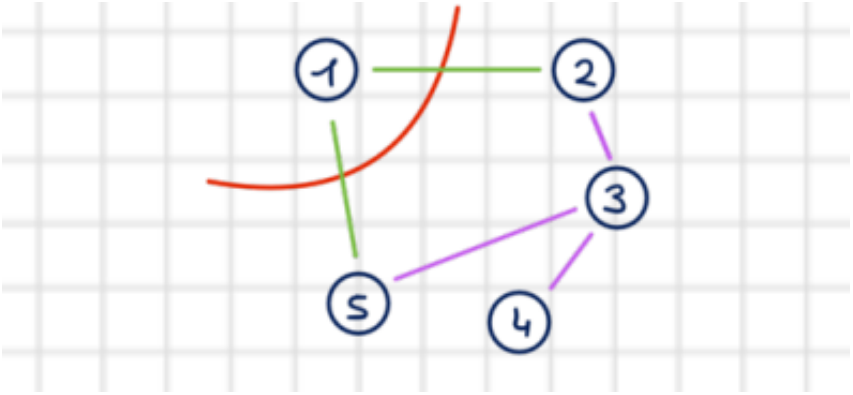

Figura 2.7: 1-Albero

> **Domanda da esame:** Ogni ciclo Hamiltoniano è un k-Albero (con grado 2 per ogni nodo), ma non viceversa. Se il k-Albero di costo minimo trovato è anche un ciclo Hamiltoniano, allora è la soluzione ottima.

2.6.2.1 Modello

$$\begin{cases} \min \sum_{i,j} c_{ij} x_{ij} \\ \sum_{i<j} x_{ij} + \sum_{j<i} x_{ji} = 2 \quad \forall j \qquad 1. \\ \sum_{i \in S | j \notin S | i<j} x_{ij} + \sum_{i \notin S | j \in S | j<i} x_{ji} \geq 1 \qquad 2. \end{cases}$$

- (1.) Per un ciclo hemiltoniano, il numero di archi collegati ad un nodo deve essere 2, il nodo *ha grado 2*. Sono detti *vincoli di grado*, sono n vincoli($n = |nodi|$), in ogni vincolo $n - 1$ legami.
- (2.) Sto sommando $x_{13} + x_{15} + x_{16} + x_{23} + x_{25} + x_{26} + x_{34} + x_{45} + x_{46} \qquad x_{ij} \in \{0, 1\}$, sono detti *vincoli di connessione*.

> I dati dell'esercizio 2 del compito sono in C, quindi per questo modello variano giusto i nodi, devo salvarmi la matrice per matlab(`intlinprog`). Se $A = 10 \times 10$ $A_{eq} = 5 \times 10$, una riga di A_{eq} ha 6 zeri. Le 10 righe variano al avaraire di S, ed S=(12,13,14,15,23,24,25,34…).

$$prima\ riga\ S = \{1,2\} | S = \{3,4,5\} \left(x_{13} + x_{14} + x_{15} + x_{23} + x_{24} + x_{25} \geq 1 \right)$$

Notare che la riga sarebbe la stessa per entrambi i sottinsiemi, infatti, anche se le righe dovrebbero essere $2^n - 2n - 2 = 20$ *in realtà si dimezzano, la riga del secondo sottinsieme avrebbe semplicemente gli indici invertiti*

> Visualizza appunti originali per domande più avanzate

2.7 7. Metodo del Branch and Bound (B&B)

È una tecnica di enumerazione intelligente che esplora l'albero delle possibili soluzioni in modo sistematico, evitando di analizzare rami che non possono portare a una soluzione migliore di quella già trovata(*incumbent*).

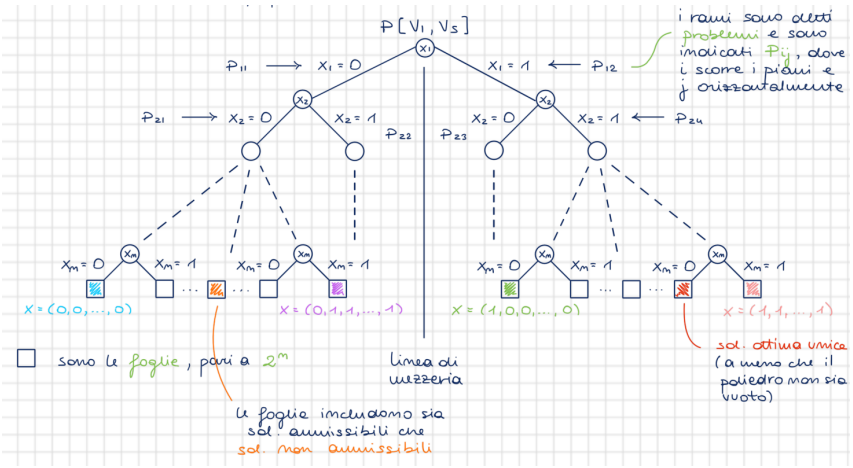

Figura 2.8: Albero B&B

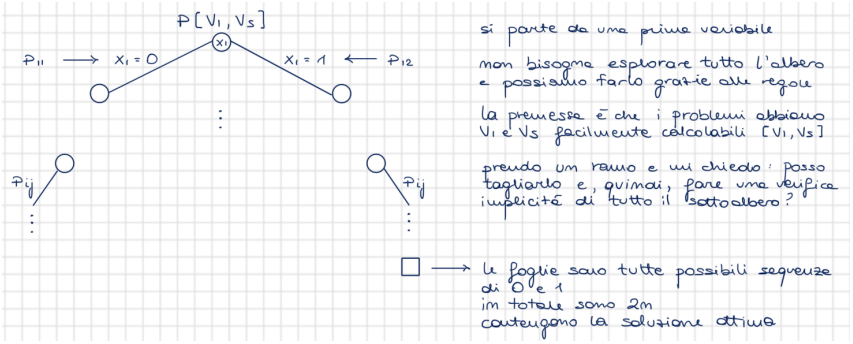

Figura 2.9: Albero B&B

Ad ogni nodo dell'albero di ricerca, che rappresenta un sotto-problema con alcune variabili fissate (es. $x_1 = 0$, i vincoli dei nodi superiori si ripecuotono sui nodi inferiori, in P31 ad esempio si hanno i vincoli x1=0 di P11 ed x2=0 di P21), si applicano tre regole di **potatura (pruning)** o **taglio**:

Regola di Taglio	Problemi di Minimizzazione (es. TSP)	Problemi di Massimizzazione (es. Zaino)
1) Taglio per Inammissibilità	Si taglia un nodo se il sotto-problema corrispondente non ha soluzioni ammissibili (è "vuoto").	Problemi in ordine di rendimento, scelgo se prendere o no quel bene, ramo sx no ramo dx si. Taglio uguale al caso di minimizzazione. Quando si imposta il bene x_i del problema che si sta studiando, le variabili della soluzione precedente rimangono ad 1, si satura con una frazione del,o direttamente col, prossimo bene
2) Taglio per Ottimalità (Bound)	Si taglia un nodo se la sua **valutazione inferiore** (V_i, es. costo dell'1-albero) è peggiore (maggiore o uguale) della migliore soluzione ammissibile trovata finora (V_s, l'incumbent). $V_i(\text{nodo}) \geq V_s(\text{globale})$	Si taglia un nodo se la sua **valutazione superiore** (V_s, es. valore del rilassato continuo, saturazione) è peggiore (minore o uguale) della migliore soluzione ammissibile trovata finora (V_i, l'incumbent). $V_s(\text{nodo}) \leq V_i(\text{globale})$
3) Aggiornamento (Taglio per Soluzione)	Se si trova una **soluzione ammissibile** (es. l'1-albero è un ciclo H.) in un nodo il cui costo è migliore (minore) dell'incumbent, questo diventa il **nuovo incumbent** ($V_s(\text{globale})$ si aggiorna). Il nodo viene quindi tagliato.	Se si trova una **soluzione ammissibile** (es. il rilassato dello zaino ha componenti intere) il cui valore è migliore (maggiore) dell'incumbent, questo diventa il **nuovo incumbent** ($V_i(\text{globale})$ si aggiorna). Il nodo viene quindi tagliato.

> **Domanda da esame:** A ogni passo del B&B per il TSP, devo ricalcolare un intero ciclo Hamiltoniano? No, si calcola solo una nuova valutazione inferiore (es. un 1-Albero) per il sotto-problema e la si confronta con l'incumbent globale. Ricorda che un k-albero è un ciclo hamiltoniano se la somma dei vincoli di grado fa 2.

2.8 8. Problema di Copertura (Set Covering)

Dato un insieme di "requisiti" (es. quartieri da servire) e un insieme di "attività" (es. stazioni di ambulanze) con i rispettivi costi, l'obiettivo è scegliere le attività a costo minimo che garantiscano la copertura di tutti i requisiti.

2.8.1 Formulazione del Modello

- **Modello Generale (Costo Minimo):**
- 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

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

2.8.2 Regole di Riduzione del Problema

Prima di risolvere, si possono applicare delle semplificazioni:

- colonna tutti 0, posso rimuovere quella variabile e ridurre il problema
- colonna tutti 1, se il problema è minor numero è gia risolto
- due righe uguali, ne tolgo una
- riga di tutti 0(problema vuoto), si elimina
- riga di tutti 1, posso eliminare, non crea problemi
- riga con un solo 1, ad esempio la riga 5, la elimino e pongo $x_5 = 1$, ovvero il quartiere 5 è servito, ma così posso rimuovere tutte le righe con x_5 posto a 1. Quella variabile sparisce dal problema.
- Regola di *dominanza*, supponiamo di avere due colonne, la colonna k domina la r se dove r è 1 anche k lo è, quindi possiamo eliminare r $A_k > A_r$. Applicabile solo in mancanza di costi.

2.8.3 Limiti v_i e v_s

- v_i **(Lower Bound):** Si ottiene risolvendo il **rilassamento continuo** del problema.

$$\begin{cases} \min \ c^T x \\ Ax \geq \vec{e} \\ x_i \in \{0, 1\} \end{cases}$$

  dove $A_{ij} = 1$ se l'attività j copre il requisito i, e $\vec{e}$ è un vettore di tutti 1.

- v_s **(Upper Bound):** S.A. trovata con algoritmi euristici.

  - **Algoritmo Greedy (caso senza costi):**
    1. Ad ogni passo, scegli l'attività che copre il maggior numero di requisiti *non ancora coperti*.
    2. Aggiungila alla soluzione e rimuovi i requisiti che hai appena coperto(colonna e tutte le righe servite).
    3. Ripeti finché tutti i requisiti sono coperti.
  - **Algoritmo di Chvatal (caso con costi):** Simile al precedente, ma ad ogni passo si sceglie l'attività con il **minimo costo unitario**, definito come:

$$u_j = \frac{c_j}{\text{numero di requisiti coperti da } j}$$

2.8.4 Problema di Massima Copertura

Variante in cui si ha un limite per un numero fisso p di attività da scegliere, e l'obiettivo è massimizzare il "valore" dei requisiti coperti (es. popolazione servita).

- **Modello di Massima Copertura:**

$$\begin{cases} \max\ h^T z \\ Ax \geq z \\ \sum x_i = p \\ x, z \in \{0,1\} \end{cases}$$

  dove $z_i = 1$ se il requisito i è coperto, e h è il vettore dei valori dei requisiti.

3 Riassunto Programmazione Lineare su Reti (PLR)

Date Importanti

- **05/11/2024:** Introduzione al Flusso di Costo Minimo in reti non capacitate, matrice di incidenza e caratterizzazione delle basi.
- **06/11/2024:** Duale del flusso di costo minimo (problema dei potenziali), calcolo dei potenziali e test di ottimalità (Teorema di Bellman).
- **11/11/2024:** Algoritmo del Simplesso su Reti, Problema del Cammino Minimo (SPT) come caso speciale.
- **12/11/2024:** Flusso di Costo Minimo in Reti Capacitate, modello con variabili di scarto e caratterizzazione delle basi (tripartizione T,L,U).
- **13/11/2024:** Duale del modello capacitato e Teorema di Bellman per reti capacitate.
- **14/11/2024:** Algoritmo del Simplesso duale per Reti Capacitate.
- **18/11/2024:** Problema del Flusso Massimo, modello matematico e riduzione a flusso di costo minimo.
- **19/11/2024:** Teorema Max-Flow/Min-Cut e Algoritmo di Ford-Fulkerson.
- **20/11/2024:** Algoritmo di Dijkstra per il problema dei cammini minimi.

3.1 1. Concetti Fondamentali e Flusso di Costo Minimo (Non Capacitato)

La Programmazione Lineare su Reti (PLR) è una specializzazione della Programmazione Lineare (PL) applicata a problemi che possono essere modellati tramite grafi. Questo approccio è fondamentale per ottimizzare flussi di risorse, percorsi minimi e assegnazioni in vari ambiti ingegneristici e logistici.

Un grafo è una coppia di insiemi $G = (N, A)$ dove $N \subset \mathbb{N}$ è l'insieme di nodi e $A \subset N \times N$ è l'insieme di archi. Una rete è un grafo a cui vengono aggiunti attributi ai nodi o agli archi. Ad esempio, bilanci associati ai nodi, costi e capacità associati agli archi.

I problemi di PLR trovano applicazione in reti di comunicazione, logistica, gestione delle risorse e pianificazione industriale. Il loro studio si basa su modelli matematici lineari risolvibili con metodi efficienti come il Simplesso su reti e algoritmi combinatori specifici.

3.1.1 Flusso di Costo Minimo nelle Reti Non Capacitate

Siano x_{ij} le unità di flusso sull'arco $(i, j) \in A$ e c_{ij} il costo per unità di flusso. L'obiettivo è trovare il flusso di costo minimo che rispetti i bilanci dei nodi.
Un nodo i è definito:

- **Sorgente/Produttore** se il suo bilancio $b_i < 0$ (flusso che esce dal nodo).
- **Pozzo/Consumatore** se il suo bilancio $b_i > 0$ (flusso che entra nel nodo).
- **Di transito** se il suo bilancio $b_i = 0$. Affinché una rete sia bilanciata, la somma di tutti i bilanci deve essere zero: $\sum_{i \in N} b_i = 0$.

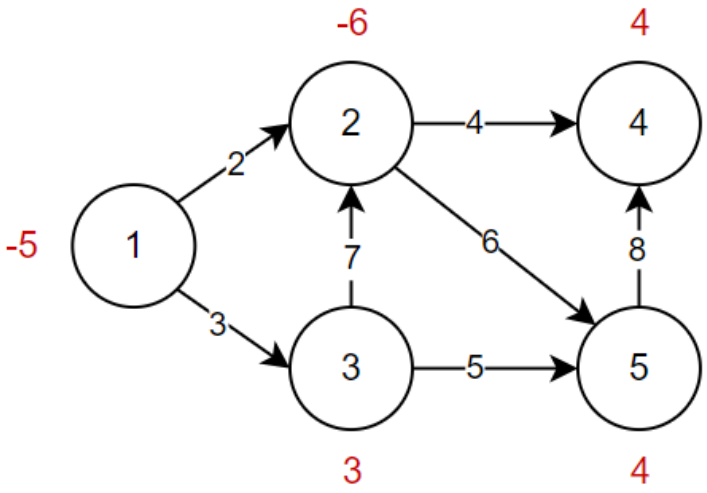

Figura 3.1: Esempio di Rete con Costi e Flussi

Modello Matematico:

$$(P)\begin{cases} \min \sum_{(i,j)\in A} c_{ij}x_{ij} & \text{(funzione obiettivo: costo totale)} \\ \sum_{j \text{ t.c. } (j,i)\in A} x_{ji} - \sum_{k \text{ t.c. } (i,k)\in A} x_{ik} = b_i \quad \forall i \in N & \text{(equazioni di bilancio per ogni nodo } i\text{)} \\ x_{ij} \geq 0 & \text{(vincolo di non negatività del flusso)} \end{cases}$$

Noto che $E \cdot \vec{x} = \vec{b}$ sono le equazioni di biliancio

$$E\begin{pmatrix} x_{12} \\ x_{13} \\ x_{24} \\ x_{25} \\ x_{32} \\ x_{35} \\ x_{54} \end{pmatrix} = \begin{pmatrix} b_1(-5) \\ b_2(-6) \\ b_3(3) \\ b_4(4) \\ b_5(4) \end{pmatrix} \quad vett.\ dei\ bilanci$$

I termini delle eq. di bilancio possono essere permutati ottenendo $\sum x_{pi} - \sum x_{iq}$

- **Modello**

$$(p)\begin{cases} min \sum\limits_{(i,j)\in A} c_{ij}x_{ij} \\ \sum\limits_{(p,j)\in A} x_{pi} - \sum\limits_{(i,q)\in A} x_{iq} = b_i \\ x \geq 0 \end{cases} \equiv \begin{cases} \min\ C^T x \\ Ex = b \\ x \geq 0 \end{cases}$$

dove E è la **matrice di incidenza della rete**.

3.1.2 Matrice di Incidenza della Rete (E)

La matrice E ha dimensioni $n \times m$ (nodi $\times$ archi). Per ogni arco (i, j):

- La riga i (sorgente dell'arco) contiene -1.
- La riga j (destinazione dell'arco) contiene $+1$.
- Tutte le altre righe nella stessa colonna contengono 0. Ogni colonna della matrice E contiene al massimo un 1 e un -1.

$$E = \begin{pmatrix} & \mathbf{12} & \mathbf{13} & \mathbf{24} & \mathbf{25} & \mathbf{32} & \mathbf{35} & \mathbf{54} \\ 1| & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 2| & 1 & 0 & -1 & -1 & 1 & 0 & 0 \\ 3| & 0 & 1 & 0 & 0 & -1 & -1 & 0 \\ 4| & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 5| & 0 & 0 & 0 & 1 & 0 & 1 & -1 \end{pmatrix}$$

La somma delle righe della matrice E è un vettore nullo, il che implica che il rango di E è $n - 1$. Questo significa che una delle equazioni di bilancio è linearmente dipendente dalle altre e può essere eliminata senza alterare il problema.

3.1.3 Teorema dell'Interezza

La matrice di incidenza E è **totalmente unimodulare**. Questo significa che ogni sua sottomatrice quadrata ha determinante $+1$ o -1. **Implicazione:** Se i bilanci b sono interi, allora le soluzioni di base (vertici del poliedro) del problema $Ex = b, x \geq 0$ sono sempre a **componenti intere**. Di conseguenza, il problema di Programmazione Lineare (PL) è equivalente al suo corrispettivo intero (PLI) e può essere risolto efficientemente con algoritmi standard di PL come il metodo del Simplesso. I problemi di trasporto e assegnamento sono, con le giuste accortezze, casi particolari del flusso di costo minimo e beneficiano di questa proprietà.

3.1.4 Caratterizzazione delle Basi

Una soluzione di base per un problema di flusso su reti è generata da un **albero di copertura** (*spanning tree*) della rete. Un albero di copertura T è un sottografo che connette tutti i nodi utilizzando esattamente $n - 1$ archi e senza formare cicli. Gli archi della rete sono partizionati in:

- T: Archi di base (quelli che formano l'albero di copertura).
- L: Archi non di base (quelli non inclusi nell'albero), il cui flusso x_{ij} è posto a 0.

> **Come controllare se una soluzione $\bar{x}$ è di base?**
>
> 1. Deve esistere un albero di copertura T tale che tutti gli archi con flusso non nullo appartengano a T.
> 2. Il numero di zeri nella soluzione deve essere almeno $m - (n - 1)$, dove m è il numero di archi e n è il numero di nodi. Se ci sono più di $m - (n - 1)$ zeri (cioè un arco in T ha flusso nullo), la soluzione di base è **degenere**.
> 3. Gli archi con flusso non nullo (quelli in T) non devono formare un ciclo.

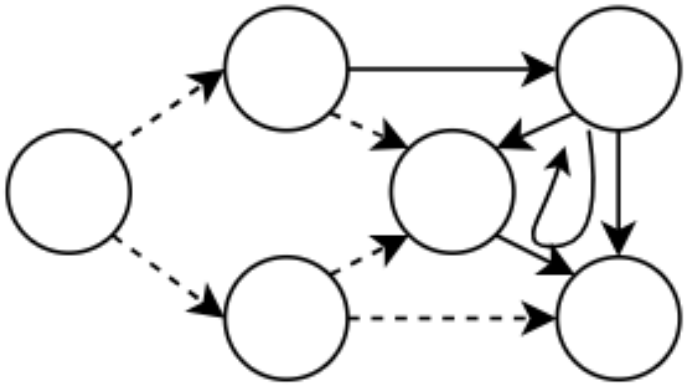

Figura 3.2: Esempio di rete con archi di base e non di base

- $m - (n-1) = 9 - 5 = 4$ archi non di base
- T **non** è un albero di copertura

3.1.5 Algoritmo per trovare un Flusso di Base (Soluzione Ammissibile)

Per trovare un flusso di base (che è un vertice del poliedro), si può effettuare una visita **posticipata per foglie** sull'albero di copertura.

1. Poni $x_{ij} = 0 \quad \forall (i,j) \in L$ (archi non di base).
2. Inizia da una foglia j dell'albero di copertura (un nodo con un solo arco incidente) e risali al nodo precedente i attraverso l'unico arco (i,j) incidente su j.
3. Assegna il flusso x_{ij} pari al bilancio b_j della foglia (o al bilancio residuo, se il nodo è già stato parzialmente servito da altri archi).
4. Elimina virtualmente l'arco (i,j) e la foglia j.
5. Aggiorna il bilancio del nodo i tenendo conto del flusso appena assegnato: $b_i := b_i - x_{ij}$ se il flusso è uscente da i, o $b_i := b_i + x_{ij}$ se il flusso è entrante in i.
6. Ripeti i passi 2-5 finché tutti i nodi non sono stati visitati e i bilanci soddisfatti.

Questo processo garantisce che la matrice di incidenza associata all'albero di copertura (E_T) sia triangolare inferiore e abbia determinante ± 1, il che assicura che la soluzione di base sia a componenti intere.

3.2 2. Duale, Test di Ottimalità e Simplesso su Reti (Non Capacitato)

3.2.1 Duale del Flusso di Costo Minimo (Problema dei Potenziali)

Il problema duale del flusso di costo minimo (che in forma primale è $\min c^T x$ soggetto a $Ex = b, x \geq 0$) è formulato come:

$$(D) \begin{cases} \max\ b^T \pi \\ E^T \pi \leq c \end{cases}$$

dove π è il vettore delle variabili duali, chiamate **potenziali** dei nodi. La dimensione di π è $n \times 1$ (uguale al numero di nodi), e le sue componenti sono non ristrette nel segno. La condizione $E^T\pi \leq c$ si traduce in una disuguaglianza per ogni arco $(i, j) \in A$:

$$-\pi_i + \pi_j \leq c_{ij} \quad \forall (i,j) \in A$$

3.2.2 Calcolo dei Potenziali di Base e Costi Ridotti

Data una base T (un albero di copertura), il **potenziale di base** $\bar{\pi}$ associato si calcola imponendo che i vincoli duali corrispondenti agli archi in T siano soddisfatti all'uguaglianza (per le condizioni di complementarità del Simplesso):

$$-\bar{\pi}_i + \bar{\pi}_j = c_{ij} \quad \forall (i,j) \in T$$

Questo è un sistema di $n-1$ equazioni in n incognite. Per trovare una soluzione unica, si fissa arbitrariamente il potenziale di un nodo, ad esempio $\pi_1 = 0$ (o un altro nodo di riferimento).

Il **costo ridotto** di un arco (i, j) rispetto a un dato potenziale π è definito come:

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

Per gli archi di base (quelli in T), i costi ridotti sono sempre zero per costruzione.

3.2.3 Teorema di Bellman (Condizione di Ottimalità)

Un flusso di base ammissibile $\bar{x}$ è **ottimo** se e solo se le **condizioni di Bellman** sono verificate per tutti gli archi non di base (quelli in L), ovvero:

$$c_{ij}^{\pi} \geq 0 \quad \forall (i,j) \in L$$

Questo equivale a dire che il potenziale di base $\bar{\pi}$ è una soluzione ammissibile per il problema duale. Se il costo ridotto di un arco non di base è 0, l'ottimo potrebbe non essere unico (la soluzione duale è degenere).

3.2.4 Algoritmo del Simplesso su Reti (Non Capacitato)

L'algoritmo del Simplesso per reti è una versione specializzata del metodo del Simplesso generale, che sfrutta la struttura della rete per maggiore efficienza. Non si invertono matrici, ma si fanno solo confronti e aggiornamenti localizzati.

Partenza: Si inizia da un albero di copertura T che genera un flusso di base ammissibile $\bar{x} = (\bar{x}_T, \bar{x}_L)$, con $\bar{x}_T \geq 0$ e $\bar{x}_L = 0$.

1. **Test di Ottimalità:** Calcola il potenziale di base $\bar{\pi}$ e i costi ridotti $c_{ij}^{\bar{\pi}}$ per tutti gli archi non di base (in L).
   - **Se** $c_{ij}^{\bar{\pi}} \geq 0 \quad \forall (i,j) \in L$, **allora** la soluzione corrente $\bar{x}$ è ottima. STOP.
   - **Altrimenti**, esiste almeno un arco $(p, q) \in L$ tale che $c_{pq}^{\bar{\pi}} < 0$. Questo è l'**arco entrante**. Per la regola anticiclo di Bland, si sceglie il primo di questi archi in ordine lessicografico.
2. **Ciclo di Compensazione:** L'aggiunta dell'arco entrante (p, q) all'albero di copertura T crea un unico **ciclo** C.
   - Orienta il ciclo in maniera concorde all'arco entrante (p, q).
   - Dividi gli archi del ciclo in due sottinsiemi:
     - C^+: archi concordi al verso di (p, q) nel ciclo.
     - C^-: archi discordi rispetto al verso di (p, q) nel ciclo.

3. **Calcolo di θ (quantità di flusso da spostare):** Calcola la massima quantità di flusso θ che può essere inviata lungo il ciclo senza violare i vincoli di non negatività. Questa è limitata dal flusso minimo presente sugli archi in C^-.

$$\theta = \min\{\bar{x}_{ij} \mid (i,j) \in C^-\}$$

   - **Se** $C^- = \emptyset$ (il ciclo è composto solo da archi concordi), allora $\theta = +\infty$. In questo caso, il problema è **illimitato** e la funzione obiettivo tende a $-\infty$. STOP.

4. **Arco Uscente:** L'arco uscente è il primo arco $(r,s) \in C^-$ che realizza il minimo θ (cioè $\bar{x}_{rs} = \theta$). Questo arco esce dalla base T ed entra in L (il suo flusso diventerà 0).

5. **Aggiornamento del Flusso:** Aggiorna il flusso corrente $\bar{x}$ per ottenere il nuovo flusso di base $x(\theta)$:

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

   La funzione obiettivo decresce di $\theta \cdot |c^{\bar{\pi}}_{pq}|$ ad ogni passo.

6. **Nuova Base:** Aggiorna la partizione (T, L) scambiando l'arco entrante con l'arco uscente. Torna al passo 1.

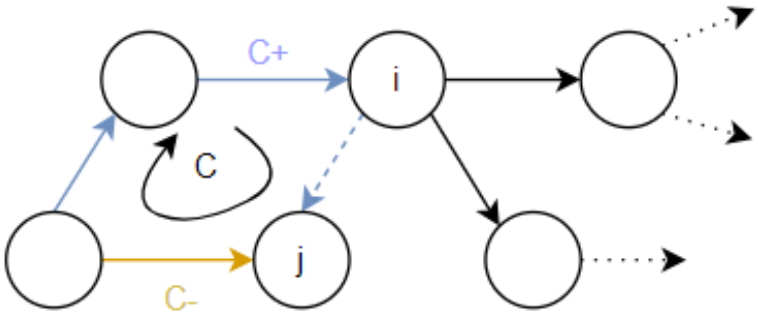

Figura 3.3: esempio

Ma per quali θ è ammissibile $x(\theta)$?

> Se chiedo se il flusso di una rete è ammissibile, guardate i bilanci, controllate se tutti i vincoli sono rispettati, il flusso deve essere positivo.
> Per il flusso di base invece? Il flusso di base lo calcolo solo sull'albero di copertura

Controlliamo quindi se è ammissibile, se lo è rispetta i bilanci $E_T x_T + \underbrace{E_L x_L}_{=0} = b$ e $x(\theta) \geq 0$:

- Bilanci(sui nodi del ciclo): ho le seguenti 4 possibilità:
  - *Nodo non appartenente al ciclo*; incidono archi non appartenenti al ciclo, non ho modificato il flusso, rispetto le condizioni $\forall\theta$
  - 1. $+\theta \to O \to +\theta$ concordi
  - 2. $-\theta \leftarrow O \to +\theta$ si compensa
  - 3. $+\theta \to O \leftarrow -\theta$ si compensa
  - 4. $-\theta \leftarrow O \leftarrow -\theta$ discordi

Quindi è ammissibile $\forall\theta$? non proprio, al momento abbiamo verificato che lo è rispetto ai bilanci $\forall\theta$, dobbiamo verificare che il flusso sia positivo:

1. Per nodi non appartenenti al ciclo $\forall\theta$
2. Su archi di C^+ $\forall\theta$
3. Su archi di C^- devo sottarre il massio θ fin quando $x(\theta) \geq 0$

in $\bar{x}_{ij} - \theta \quad (i,j) \in C^-$ rischio che non sia positivo, quanto posso sottrarre affinchè non vada oltre?

- $\bar{\theta} \leq \min\{\bar{x}_{ij}\}$ $(i,j) \in C^-$, ma so che maggiore è θ più la funzione decresce, quindi è più corretto:
- $\bar{\theta} = \min\{\bar{x}_{ij}\}$ $(i,j) \in C^-$, uno degli archi ora si è svuotato, ed andrà in L(dove sono gli archi nulli)

> Per la regola anticiclo di Bland, prendo il primo di questi archi: questo è l'arco uscente. Poi ricalcolo il nuovo potenziale, vedo se non è ammissibile, in tal caso eseguo un altro passo del simplesso.

3.2.5 Esempio: Soluzione di Base Tramite Ispezione Diretta

Per capire se una soluzione $\bar{x}$ è di base, è necessario che il numero di variabili non nulle sia $n-1$ e che queste non formino cicli, garantendo che costituiscano un albero di copertura.

> **Esempio Soluzione Non Ammissibile e Non Degenere**
>
> $$x = \begin{pmatrix} 12 & 13 & 24 & 25 & 32 & 35 & 54 \\ -2 & 7 & 4 & 0 & 0 & 4 & 0 \end{pmatrix}$$
>
> Questa soluzione non è ammissibile perché $x_{12} = -2 < 0$. Non è degenere perché non ci sono archi in T con flusso zero oltre agli archi non di base (che per definizione hanno flusso zero).

3.3 3. Problema del Cammino Minimo (SPT)

Il problema del **Cammino Minimo** (Shortest Path) è uno dei problemi più noti e polinomialmente risolvibili sulle reti. L'obiettivo è trovare il cammino di costo minimo da un nodo sorgente r a un nodo destinazione q, o a tutti gli altri nodi.

3.3.1 SPT come Flusso di Costo Minimo

Il problema SPT può essere modellato come un problema di flusso di costo minimo non capacitato, definendo opportuni bilanci dei nodi.
Per trovare il cammino minimo da un nodo r a un nodo q:

- $b_r = -1$ (sorgente del flusso)
- $b_q = 1$ (pozzo del flusso)
- $b_i = 0$ per tutti gli altri nodi $i \neq r, q$ Questo problema è risolvibile con il Simplesso su reti.

Per trovare tutti i cammini minimi da un nodo r a tutti gli altri $n-1$ nodi della rete:

$$b_i = \begin{cases} -(n-1) & \text{se } i = r \\ 1 & \text{se } i \neq r \end{cases}$$

Questa formulazione genera un flusso tale che ogni nodo riceve una unità di flusso dalla sorgente r attraverso un cammino minimo. La soluzione ottima di questo problema è un albero di copertura, chiamato **Shortest Path Tree (SPT)**, che contiene tutti i cammini minimi dalla radice.

Il **modello** è

$$\begin{cases} \min\ c^T x \\ Ex = b : b_i = \begin{cases} -(n-1), & i = r \\ 1, & i \neq r \end{cases} \\ x \geq 0 \end{cases}$$

Proprietà Fondamentali:

1. **Non Degenerazione:** Il problema dei cammini minimi **non ammette soluzioni di base degeneri**. Il flusso su ogni arco dell'albero di base è un intero strettamente positivo, pari al numero di nodi nel sottoalbero servito da quell'arco. Pertanto, la regola anticiclo di Bland non è necessaria in questo contesto.
2. **Arco Uscente:** Dato un arco entrante (p, q), l'arco uscente deve essere necessariamente l'unico arco in T che entra nel nodo q.

3.3.1.1 Simplesso per cammini minimi

1. Trovo un albero T orientato di radice r.

2. Calcolo il potenziale di base $\bar{\pi}^T := c_T^T E_T^{-1}$.
   Indico con $c_{ij}^{\bar{\pi}} := c_{ij} + \bar{\pi}_i - \bar{\pi}_j$ il costo ridotto $\forall (i, j)$.

3. **if** le condizioni di Bellman sono verificate, i.e.
   $c_{ij}^{\bar{\pi}} \geq 0\ \forall (i, j) \notin T$
   then STOP (T è un albero dei cammini minimi).

   else scelgo come arco entrante un arco (p, q) tale che $c_{pq}^{\bar{\pi}} < 0$
   (possibilmente il più piccolo costo ridotto).

4. L'arco (p, q) forma un ciclo C con gli archi di T.

   if tutti gli archi di C sono concordi con (p, q)
   then STOP (non esiste albero dei cammini minimi e C è un ciclo di costo negativo).

   else scelgo l'arco uscente come l'unico arco $(i, q) \in T$ che entra nel nodo q.

5. Aggiorno l'albero e torno al passo 2.

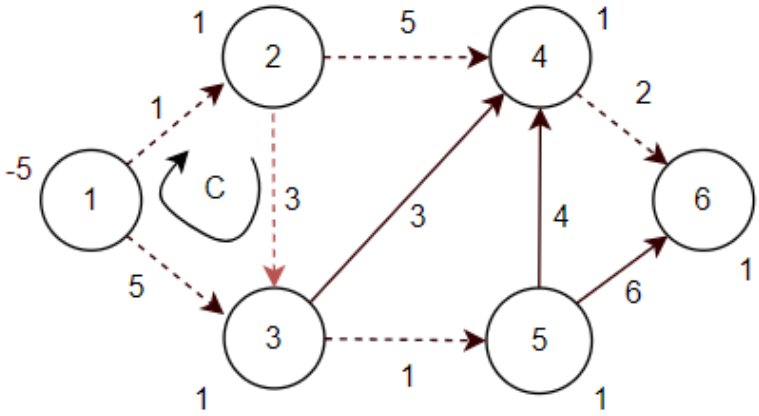

Figura 3.4: Esempio

Dato il vettore dei potenziali di Bellman $\overline{\pi} = (0, 1, 5, 6, 6, 8)$, si nota che questo non rappresenta l'albero dei cammini minimi. Per raggiungere il nodo (3), è più efficiente passare per il nodo (2).

L'arco che viola la condizione di Bellman è (2, 3). Infatti, il costo ridotto c_{23}^{π} è calcolato come:

$c_{23}^{\pi} = 3 + 1 - 5 = -1$

Di conseguenza, l'arco uscente è necessariamente (1, 3).

La funzione obiettivo diminuisce di $5 - 3 - 1 = 1$ per ogni cammino che include l'arco (1, 3). La diminuzione totale è pari al flusso sull'arco (1, 3), indicato come $\overline{x}_{13}$. In questo caso specifico, il flusso è pari a 2.

> *Pagina 99 della dispensa*, trasporto di costo minimo come flusso su reti e altri esempi/domande

---

3.4 4. Flusso di Costo Minimo su Reti Capacitate

Nelle reti capacitate, oltre al costo c_{ij}, a ciascun arco è associata una **capacità massima** u_{ij}. Il flusso x_{ij} su un arco deve rispettare $0 \leq x_{ij} \leq u_{ij}$.

3.4.1 Simboli e Notazione Utilizzati

- T: insieme di archi **nella base attuale** (arco attivo nella soluzione di base).
- L: insieme degli archi **a limite inferiore di flusso**.
- U: insieme degli archi **a limite superiore di flusso**.
- u_{kl}: **capacità massima** dell'arco kl.
- x_{kl}: **flusso attuale** sull'arco kl.
- θ: **variazione di flusso** calcolata per equilibrare il ciclo generato dall'arco entrante.
- C^{+}: insieme degli archi **concordi** con l'orientamento del ciclo.
- C^{-}: insieme degli archi **discordi** rispetto all'orientamento del ciclo.

3.4.2 Modello Matematico con Variabili di Scarto

Per includere il vincolo di capacità e portare il problema in una forma più standard, si introducono le **variabili di scarto** $w_{ij} \geq 0$ per ogni arco:

$$x_{ij} + w_{ij} = u_{ij} \quad \forall (i,j) \in A$$

dove w_{ij} rappresenta la capacità residua non utilizzata sull'arco.

Il problema si riformula come:

$$\begin{cases} \min\ c^T x + \mathbf{0}^T w \\ Ex = b \\ x + w = u \\ x, w \geq 0 \end{cases}$$

La matrice dei vincoli del poliedro dei flussi in reti capacitate diventa:

$$\begin{bmatrix} E^T & I \\ O & I \end{bmatrix}^T \begin{pmatrix} x \\ w \end{pmatrix} = \begin{pmatrix} b \\ u \end{pmatrix}$$

Questa matrice ha dimensioni $(n-1+m) \times 2m$. Il suo rango è $m+n-1$. Anche in questo caso, la matrice è totalmente unimodulare, e quindi le soluzioni di base sono a componenti intere se i bilanci e le capacità sono interi.

3.4.3 Caratterizzazione delle Basi (Reti Capacitate)

Nelle reti capacitate, una base è definita da una **tripartizione** degli archi $A = T \cup L \cup U$:

- T: Insieme degli archi **nell'albero di copertura** (di base). Il loro flusso x_{ij} deve essere strettamente compreso tra 0 e u_{ij} (non vuoto e non saturo).
- L: Insieme degli archi **a limite inferiore di flusso** (non di base). Il loro flusso è $x_{ij} = 0$.
- U: Insieme degli archi **a limite superiore di flusso** (non di base), detti anche archi **saturi**. Il loro flusso è $x_{ij} = u_{ij}$.

Il sottoinsieme T deve contenere esattamente $n-1$ archi, formando un albero di copertura. Gli altri $m-(n-1)$ archi sono distribuiti tra L e U. Una soluzione di base è **degenere** se almeno una componente di base è zero. In questo contesto, ciò significa che un arco in T ha flusso $x_{ij} = 0$ o $x_{ij} = u_{ij}$ (cioè, è vuoto o saturo).
Anche le righe relative a w, essendo variabili di scarto legate agli archi, devono essere suddivise analogamente. Si ottiene quindi una *esapartizione* della matrice del poliedro dei flussi su reti capacitate nei seguenti blocchi:

- T, L, U, T', L', U'.

Per ottenere una **soluzione di base ammissibile**, si cerca una tripartizione che soddisfi questa condizione. Finché non si ha a disposizione un metodo ausiliario, si possono scegliere casualmente le tripartizioni, selezionando un albero di copertura per T e assegnando gli archi rimanenti in L o U.

La soluzione di base si costruisce ricordando che:

$$w_{ij} = u_{ij} - x_{ij}$$

Le condizioni sono quindi:

- $x = u$ per gli archi saturi in U
- $w = u$ per gli archi in L', poiché $w_{L'} = u - x$

- i flussi x_T e le variabili $w_{T'}$ devono essere determinati per gli archi nell'albero di copertura.

Una soluzione è ammissibile se tutte le componenti (x, w) sono maggiori o uguali a zero:

- Se $w_{ij} < 0$, allora $x_{ij} > u_{ij}$, violando il vincolo di capacità sull'arco.

$$(x, w) = \begin{pmatrix} x_T & x_L & x_U & w_{T'} & w_{L'} & w_{U'} \\ ? & 0 & u_U & u_T - x_T & u_L & 0 \end{pmatrix}$$

Una soluzione di base è **degenere** se almeno una componente della base è zero. Tuttavia, poiché la portata deve essere maggiore di zero, non ci possono essere componenti nulle in U né in L'. Una soluzione di base è quindi degenere se uno degli archi di T è vuoto o saturo. Pertanto, per verificare se una soluzione è degenere, **è sufficiente esaminare** x_{ij}, senza necessità di calcolare w_{ij}.

Come si costruisce una soluzione di base ammissibile? Si seleziona un albero di copertura T. Gli archi non in T vengono arbitrariamente assegnati a L (flusso 0) o a U (flusso u_{ij}). I flussi sugli archi in T vengono poi determinati risolvendo il sistema di bilanci. Una soluzione è ammissibile se tutti i flussi x_{ij} soddisfano $0 \leq x_{ij} \leq u_{ij}$.

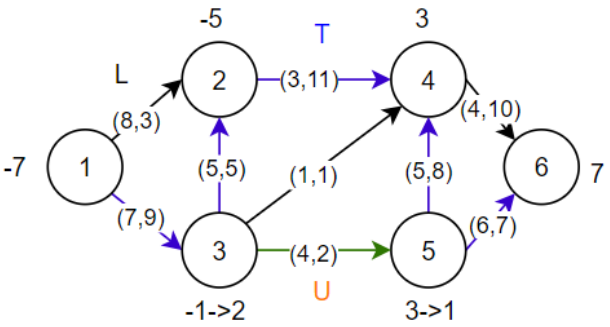

Figura 3.5: Esempio di Flusso di Base Non Ammissibile e Degenerato

	12	13	24	32	34	35	45	46	56
x	0	7	11	6	0	2	8	0	7
w	3	2	0	-1	1	0	0	10	0

Nell'esempio, la soluzione $x_{32} = 6$ supera la capacità $u_{32} = 5$, quindi $w_{32} = -1$ e la soluzione non è ammissibile.

3.5 5. Duale e Simplesso su Reti Capacitate

3.5.1 Duale del Modello Capacitato (Problema dei Potenziali)

Il problema duale del flusso di costo minimo su reti capacitate è formulato come:

$$(D_{cap}) \begin{cases} \max\ b^T\pi + u^T\mu \\ E^T\pi + I\mu \leq c \\ \mu \leq 0 \end{cases}$$

dove:

- $\pi \in \mathbb{R}^n$ sono i potenziali dei nodi e le sue componenti sono non ristrette nel segno.
- $\mu \in \mathbb{R}^m$ sono i moltiplicatori duali associati ai vincoli di capacità ($x_{ij} \leq u_{ij}$).

Le condizioni di ammissibilità duale $E^T\pi + \mu \leq c$ si traducono in:

$$-\pi_i + \pi_j + \mu_{ij} \leq c_{ij} \quad \forall (i,j) \in A$$

3.5.2 Calcolo del Potenziale di Base

Data una tripartizione (T, L, U) che definisce una base, il **potenziale di base associato** $(\bar{\pi}, \bar{\mu})$ si calcola imponendo che i vincoli duali siano soddisfatti all'uguaglianza per gli archi in T e U, e che $\mu_{ij} = 0$ per gli archi in T e L (per complementarità: se $0 < x_{ij} < u_{ij}$, allora $\mu_{ij} = 0$). Da cui si ottiene il sistema:

$$\begin{cases} -\pi_i + \pi_j = c_{ij} & \forall (i,j) \in T \\ \mu_{ij} = 0 & \forall (i,j) \in T \\ -\pi_i + \pi_j + \mu_{ij} = c_{ij} & \forall (i,j) \in U \\ \mu_{ij} = 0 & \forall (i,j) \in L \end{cases}$$

Fissando un nodo (es. $\pi_1 = 0$), si risolve il sistema per trovare $\bar{\pi}$ e le $\bar{\mu}_{ij}$ per gli archi in U.

3.5.3 Condizioni di Bellman (per Reti Capacitate)

Un flusso di base ammissibile $\bar{x}$ è **ottimo** se e solo se il potenziale di base associato $(\bar{\pi}, \bar{\mu})$ è ammissibile per il problema duale. Questo si verifica se le seguenti condizioni sui costi ridotti $c^{\pi}_{ij} = c_{ij} + \pi_i - \pi_j$ sono soddisfatte per gli archi non di base:

- **Per archi in L (flusso a zero):** $c^{\pi}_{ij} \geq 0 \quad \forall (i,j) \in L$. (Se il flusso è zero, non deve essere conveniente aumentarlo).
- **Per archi in U (flusso saturo):** $c^{\pi}_{ij} \leq 0 \quad \forall (i,j) \in U$. (Se il flusso è saturo, non deve essere conveniente diminuirlo).

Un potenziale di base è **degenere** se almeno una delle disuguaglianze non di base è verificata con l'uguaglianza, cioè quando almeno uno dei costi ridotti non di base è pari a 0.

3.5.4 Algoritmo del Simplesso Duale per Reti Capacitate

L'algoritmo del Simplesso duale si applica quando la soluzione di base primale è ammissibile, ma il potenziale di base duale non lo è (cioè, una condizione di Bellman è violata). L'obiettivo è ripristinare l'ammissibilità duale mantenendo l'ottimalità.

Partenza: Una tripartizione (T, L, U) che genera un flusso di base ammissibile $\bar{x}$.

1. **Test di Ottimalità:** Calcola il potenziale di base $\bar{\pi}$ e i costi ridotti $c_{ij}^{\bar{\pi}}$ per tutti gli archi non di base (in $L \cup U$).
   - **Se** le condizioni di Bellman sono verificate ($c_{ij}^{\bar{\pi}} \geq 0 \forall (i,j) \in L$ e $c_{ij}^{\bar{\pi}} \leq 0 \forall (i,j) \in U$), **allora** la soluzione corrente $\bar{x}$ è ottima. STOP.
   - **Altrimenti**, esiste un arco violante che sarà l'**arco entrante** (p, q):
     - Un arco $(p, q) \in L$ con $c_{pq}^{\bar{\pi}} < 0$.
     - Oppure un arco $(p, q) \in U$ con $c_{pq}^{\bar{\pi}} > 0$. Si sceglie il primo in ordine lessicografico (regola di Bland) se ci sono più violazioni.
2. **Ciclo e Orientamento:** L'arco entrante (p, q) forma un ciclo C con gli archi in T.
   - **Se** $(p, q) \in L$ e $c_{pq}^{\bar{\pi}} < 0$: orienta il ciclo in maniera **concorde** a (p, q).
   - **Se** $(p, q) \in U$ e $c_{pq}^{\bar{\pi}} > 0$: orienta il ciclo in maniera **discorde** a (p, q). Dividi gli archi del ciclo in C^+ (concordi all'orientamento del ciclo) e C^- (discordi).
3. **Calcolo di θ (variazione di flusso):** Calcola la massima variazione di flusso θ che può essere applicata lungo il ciclo senza violare i limiti di capacità $[0, u_{ij}]$:
   - $\theta^+ = \min\{u_{ij} - \bar{x}_{ij} \mid (i,j) \in C^+\}$ (capacità residua sugli archi concordi)
   - $\theta^- = \min\{\bar{x}_{ij} \mid (i,j) \in C^-\}$ (flusso sugli archi discordi)
   - $\theta = \min(\theta^+, \theta^-)$
4. **Arco Uscente:** L'arco uscente è l'arco (r, s) che realizza il minimo θ:
   - **Se** $\theta = \theta^+$ (e $\theta < \theta^-$): l'arco $(r, s) \in C^+$ è quello che raggiunge la sua capacità massima. Questo arco esce da T ed entra in U (diventa saturo).
   - **Se** $\theta = \theta^-$ (e $\theta \leq \theta^+$): l'arco $(r, s) \in C^-$ è quello che raggiunge il suo limite inferiore di zero. Questo arco esce da T ed entra in L (diventa vuoto).
   - **Caso di Degenerazione ($\theta^+ = \theta^-$):** Ci sono più archi candidati ad uscire. Si applica la regola di Bland (es. scegliere il primo in ordine lessicografico) per selezionare l'arco uscente dall'unione di L ed U.
5. **Aggiornamento:** Si aggiorna il flusso $\bar{x}$ con $x(\theta)$, si aggiorna la tripartizione (T, L, U) scambiando l'arco entrante con l'arco uscente, e si torna al passo 1.

> Pag 115 della dispensa per esempi.

3.5.5 Guida Rapida Per Analisi di Flussi e Potenziali su Reti

Questa guida riassume i criteri per verificare se una soluzione di flusso $\bar{x}$ o un vettore dei potenziali π è ammissibile, di base, degenere o ottimo.

3.5.5.1 Reti Non Capacitate

3.5.5.1.1 Analisi di un Flusso x

- **Ammissibile**: Un flusso è ammissibile se:
  1. Rispetta i **bilanci** in ogni nodo: $\sum x_{entranti} - \sum x_{uscenti} = b_i$.
  2. Ha componenti **non-negative**: $x_{ij} \geq 0$ per ogni arco.

- **Di Base**: Un flusso ammissibile è di base se gli archi con flusso $x_{ij} > 0$ **non formano cicli** e sono al più $n - 1$.
- **Degenere**: Un flusso di base è degenere se un arco dell'albero di copertura di base ha un flusso nullo (ovvero, la soluzione ha più di $m - (n - 1)$ zeri).
- **Ottimo**: Un flusso di base ammissibile è ottimo se il suo potenziale associato π soddisfa le **condizioni di Bellman**: il costo ridotto $c^{\pi}_{ij} = c_{ij} + \pi_i - \pi_j$ è ≥ 0 per tutti gli archi non in base.

3.5.5.1.2 Analisi di un Vettore dei Potenziali π

- **Ammissibile**: Un potenziale è ammissibile se $\pi_j - \pi_i \leq c_{ij}$ per **tutti** gli archi (i, j) della rete.
- **Di Base**: Un potenziale è di base se è associato a un albero di copertura T e soddisfa $\pi_j - \pi_i = c_{ij}$ per tutti gli archi $(i, j) \in T$.
- **Degenere**: Un potenziale di base è degenere se soddisfa $\pi_j - \pi_i = c_{ij}$ anche per almeno un arco (i, j) **non** appartenente alla base T.
- **Ottimo**: Un potenziale è ottimo **se è ammissibile**.

3.5.5.2 Reti Capacitate

3.5.5.2.1 Analisi di un Flusso x

- **Ammissibile**: Un flusso è ammissibile se rispetta i **bilanci** ($Ex = b$) e i **vincoli di capacità** ($0 \leq x_{ij} \leq u_{ij}$).
- **Di Base**: Un flusso è di base se è generato da una **tripartizione** degli archi (T, L, U), dove T è un albero di copertura, gli archi in L hanno flusso nullo ($x_{ij} = 0$), e gli archi in U sono saturi ($x_{ij} = u_{ij}$).
- **Degenere**: Un flusso di base è degenere se un arco dell'albero T è **vuoto** ($x_{ij} = 0$) o **saturo** ($x_{ij} = u_{ij}$).
- **Ottimo**: Un flusso di base ammissibile è ottimo se i suoi potenziali associati soddisfano le **condizioni di Bellman per reti capacitate**.

3.5.5.2.2 Analisi di un Vettore dei Potenziali π

- **Ammissibile**: Un potenziale è ammissibile se soddisfa le due condizioni di Bellman per reti capacitate:
  1. $c^{\pi}_{ij} \geq 0$ per tutti gli archi a flusso nullo (in L).
  2. $c^{\pi}_{ij} \leq 0$ per tutti gli archi a flusso saturo (in U).
- **Di Base**: Un potenziale è di base se è associato a una tripartizione (T, L, U).
- **Degenere**: Un potenziale di base è degenere se una delle condizioni di Bellman per un arco in L o U è soddisfatta con l'**uguaglianza**.
- **Ottimo**: Un potenziale è ottimo **se è ammissibile**.

3.5.5.2.3 Costruzione della Soluzione dato (T, L, U) Se ti viene fornita una base (T, L, U), ecco come costruire il flusso e i potenziali corrispondenti.

- **Come costruire il flusso x**:
  1. **Imposta i flussi non di base**: Per definizione, gli archi in L hanno flusso nullo e quelli in U sono saturi.
     - Per ogni arco $(i, j) \in L$, imposta $x_{ij} = 0$.

- Per ogni arco $(i, j) \in U$, imposta $x_{ij} = u_{ij}$.

2. **Calcola i Bilanci Modificati** b': Il flusso degli archi saturi in U modifica i bilanci originali b. Per ogni nodo i, calcola un bilancio "residuo" b'_i:
$$b'_i = b_i - \left(\sum_{(k,i) \in U} u_{ki} - \sum_{(i,j) \in U} u_{ij} \right)$$
3. **Calcola i flussi di base**: Risolvi il sistema $E_T x_T = b'$ usando l'albero di copertura T e i bilanci modificati b'. Questo si fa con una visita "posticipata per foglie" sull'albero T.

- **Come costruire il potenziale di base** π:

1. **Fissa un Potenziale**: Per convenzione, imposta il potenziale di un nodo a zero (es. $\pi_1 = 0$).
2. **Risolvi per gli altri nodi**: Usa gli archi dell'albero di copertura T per trovare gli altri potenziali. Attraversando l'albero a partire dal nodo 1, risolvi il sistema di equazioni:
$$\pi_j - \pi_i = c_{ij} \quad \forall (i, j) \in T$$

3.6 6. Problema del Flusso Massimo

In una rete, dati un nodo sorgente s e un nodo pozzo t, e le capacità u_{ij} per ogni arco, il problema del Flusso Massimo consiste nel determinare la quantità massima di flusso v che può essere inviata da s a t.

3.6.1 Modello Matematico

Il problema può essere formulato come:

$$\begin{cases} \max\ v \\ \sum_{j \text{ t.c. } (j,i) \in A} x_{ji} - \sum_{k \text{ t.c. } (i,k) \in A} x_{ik} = b_i \quad \forall i \in N \\ 0 \le x_{ij} \le u_{ij} \quad \forall (i,j) \in A \end{cases} \rightarrow \begin{cases} \max\ 0x + 1v \\ Ex = b \\ 0 \le x \le u \\ 0 \le v \le \epsilon \end{cases}$$

Dove ϵ può essere 17 nel caso di prima, somma delle capacità uscenti dal nodo 1, oppure 1000 o un numero eccessivamente grande da essere accettato. E dove i bilanci b_i sono:

$$b_i = \begin{cases} -v & \text{se } i = s \\ v & \text{se } i = t \\ 0 & \text{se } i \neq s, t \end{cases}$$

La variabile v è l'incognita che rappresenta il flusso massimo.

Questo problema può essere trasformato in un problema di **flusso di costo minimo su rete capacitata** aggiungendo un **arco fittizio** (t, s) che collega il pozzo t alla sorgente s.

- Assegna a questo arco fittizio un costo $c_{ts} = -1$ e una capacità $u_{ts} = +\infty$ (o un valore molto grande).
- Assegna costo 0 a tutti gli altri archi originali.
- Imposta tutti i bilanci $b_i = 0$. Massimizzare il flusso v equivale a minimizzare il costo totale, poiché solo l'arco (t, s) ha un costo negativo, e la quantità di flusso che lo attraversa è proprio v.

Per avviare il metodo del **simplesso**:

- Si può utilizzare una soluzione iniziale in cui **tutti i flussi sono zero**, che è sempre ammissibile e non necessita calcoli.
- Questa scelta è anche **greedy**, poiché il primo arco entrante sarebbe x_{ts}.

3.6.2 Teorema Max-Flow / Min-Cut

Il teorema Max-Flow / Min-Cut stabilisce una relazione fondamentale tra il flusso massimo e i tagli di una rete:

- Un **taglio** di una rete è una partizione dei nodi N in due sottoinsiemi N_s e N_t, dove $s \in N_s$ e $t \in N_t$.
- La **capacità di un taglio** $u(N_s, N_t)$ è la somma delle capacità degli archi diretti, cioè quelli che vanno da un nodo in N_s a un nodo in N_t. Il teorema afferma che **il valore del flusso massimo da s a t è uguale alla capacità del taglio di capacità minima** tra s e t:

$$\max v = \min_{(N_s, N_t)} u(N_s, N_t)$$

Il problema del taglio di capacità minima è il duale del problema di flusso massimo.

3.6.3 Algoritmo di Ford-Fulkerson

L'algoritmo di Ford-Fulkerson è un metodo iterativo per calcolare il flusso massimo. Si basa sulla ricerca ripetuta di "cammini aumentanti" all'interno di un grafo residuo, che rappresenta la capacità di flusso ancora disponibile nella rete. L'obiettivo è spingere quanto più flusso possibile dalla sorgente `s` al pozzo `t` fino a saturare la rete.

Grafo Residuo (G_{res}):
Per ogni arco (i, j) del grafo originale con capacità c_{ij} e flusso corrente x_{ij}, il grafo residuo G_{res} è una rappresentazione dinamica della rete che include:

- Un **arco diretto** (i, j) con **capacità residua** r_{ij}, che indica quanto flusso aggiuntivo si può ancora inviare da i a j.
- Un **arco inverso** (fittizio) (j, i) con **capacità residua** r_{ji}, che rappresenta la quantità di flusso già inviato che si può "annullare" o "spingere indietro" da j a i.

Cammino Aumentante:
Un cammino aumentante è un qualsiasi percorso orientato da s a t nel **grafo residuo**, tale che tutti gli archi su questo percorso abbiano una capacità residua $r_{ij} > 0$. Non deve contenere cicli.

- **Procedura dell'Algoritmo:**

  1. **Inizializzazione:** Si parte da una situazione di flusso nullo.
     - Il flusso totale v è impostato a 0.
     - Il flusso su ogni arco x_{ij} è impostato a 0.
     - Le capacità residue iniziali sono: $r_{ij} = c_{ij}$ per gli archi diretti e $r_{ji} = 0$ per quelli inversi.
  2. **Ricerca Cammino Aumentante:** Nel grafo residuo corrente, trova un cammino aumentante P da s a t. Per questa ricerca si utilizza solitamente la **Variante di Edmonds-Karp**.
     - Se **non esiste** alcun cammino con capacità positiva, l'algoritmo **termina**. Il valore v attuale è il flusso massimo.
     - Se un cammino viene trovato, procedi al passo successivo.
  3. **Aggiornamento:** Una volta trovato un cammino:

1. Calcola la **capacità del cammino**, indicata con δ. Questo valore è pari alla **minima capacità residua** tra tutti gli archi che compongono il percorso P.

$$\delta = \min\{r_{ij} \mid (i,j) \in P\}$$

2. Aggiorna il **flusso totale** inviato dalla sorgente al pozzo:

$$v_{nuovo} = v_{vecchio} + \delta$$

3. Aggiorna il flusso lungo il cammino P: Per ogni arco $(i,j) \in P$:

- $x_{ij} := x_{ij} + \delta$ (**aumenta** il flusso sull'arco originale)
- $x_{ji} := x_{ji} - \delta$ (**riduci** il flusso sull'arco inverso, se esiste e se il cammino aumentante ha usato il verso inverso di un arco originale)

4. Aggiorna le **capacità residue** lungo il cammino per preparare l'iterazione successiva. Per ogni arco (i,j) del cammino, sposta δ unità di capacità dalla direzione percorsa a quella opposta:
   - La capacità dell'arco diretto **diminuisce** di δ: $r_{ij} := r_{ij} - \delta$.
   - La capacità dell'arco inverso **aumenta** di δ: $r_{ji} := r_{ji} + \delta$.

4. **Ripeti:** Torna al passo 2 per cercare un nuovo cammino aumentante nel grafo residuo appena aggiornato.

Variante di Edmonds-Karp:
Per garantire una maggiore efficienza e la terminazione, questa variante stabilisce che la ricerca del cammino aumentante (Passo 2) debba essere effettuata tramite una **Ricerca in Ampiezza (BFS)**. Questo assicura di trovare sempre il cammino **più corto** in termini di numero di archi. I tuoi appunti descrivono una procedura pratica per la BFS ("algoritmo della croce"):

1. **Calcolo della stella uscente**:
   - La stella uscente FS di un nodo è l'insieme dei nodi raggiungibili attraverso archi uscenti con $r_{ij} > 0$.
   - Scrivi il nodo s sulla colonna di sinistra.
   - Nella colonna di destra, elenca i nodi $j \in FS$ in ordine lessicografico.
2. **Iterazione**:
   - Prendi il primo nodo j della colonna di destra e spostalo nella colonna di sinistra.
   - Aggiungi nella colonna di destra i nodi raggiungibili da j non ancora visitati, per cui $r_{ij} > 0$.
3. **Cammino aumentante**:
   - Continua finché nella colonna di destra non appare il nodo t.
   - Risali la colonna di sinistra per ricostruire il cammino aumentante, determinando il nodo predecessore per ogni nodo.

Si avrà quindi, incorporandolo nell'algoritmo di F-F: Karp $\rightarrow \delta \rightarrow x_{new} \rightarrow r_{new}$

- **Risultato Finale e Taglio Minimo** Quando l'algoritmo termina, il valore finale di v è il **flusso massimo**. Inoltre, secondo il **teorema max-flow/min-cut**, questo valore è uguale alla capacità del taglio minimo della rete. Il taglio si identifica così:
  - N_s: L'insieme di tutti i nodi ancora raggiungibili da s nel **grafo residuo finale**.
  - N_t: L'insieme di tutti gli altri nodi non raggiungibili.

> Esempio pagina 127 della dispensa

3.7 Algoritmo di Dijkstra

L'algoritmo di Dijkstra è un algoritmo *greedy* altamente efficiente per trovare l'albero dei cammini minimi da un nodo sorgente r a tutti gli altri nodi in una rete, a condizione che i **costi degli archi siano non negativi** ($c_{ij} \geq 0$).

- **Strutture Dati Necessarie:**
  1. **Vettore delle distanze/etichette** $\pi \in \mathbb{R}^n$: $\pi[j]$ memorizza il costo minimo (o distanza) trovata finora per raggiungere il nodo j dalla sorgente r. Inizialmente $\pi[r] = 0$ e $\pi[i] = +\infty$ per $i \neq r$.
  2. **Vettore dei predecessori** $p \in \mathbb{R}^n$: $p[j]$ memorizza il nodo che precede j nel cammino minimo scoperto. Inizialmente $p[r] = 1$ e $p[i] = \emptyset$ per $i \neq r$.
  3. **Insieme dei nodi da processare** N: Contiene i nodi per i quali il cammino minimo non è stato ancora fissato. Inizialmente contiene tutti i nodi.
- **Procedura dell'Algoritmo:**
  1. **Inizializzazione:** Imposta $\pi[r] = 0$, $p[r] = r$, e per tutti gli altri nodi i, $\pi[i] = +\infty$ e $p[i] = \emptyset$. Inserisci tutti i nodi nell'insieme N.
  2. **Iterazione Principale:** Ripeti finché N non è vuoto:
     1. **Selezione Nodo Minimo:** Seleziona e rimuovi da N il nodo i con la distanza π_i minima tra quelli ancora in N. Questo π_i è ora definitivo.
     2. **Elaborazione Stella Uscente (Rilassamento):** Per ogni arco (i, j) uscente da i (cioè per ogni $j \in FS(i)$) dove j è ancora in N:
        - Se $\pi_j > \pi_i + c_{ij}$ (ovvero, è stato trovato un cammino più breve per j passando per i):
        - Aggiorna $\pi_j = \pi_i + c_{ij}$.
        - Aggiorna $p_j = i$.

L'algoritmo termina quando tutti i nodi sono stati visitati e le distanze minime calcolate. I valori finali di π corrispondono ai potenziali ottimi del problema duale del cammino minimo.

Efficienza: Dijkstra è più efficiente del Simplesso su flussi di costo minimo per questo tipo di problema perché considera solo la stella uscente del nodo selezionato e non valuta tutti i nodi a ogni passo. Il tempo di esecuzione può variare da $O(V^2)$ (con lista semplice) a $O((V + E) \log V)$ (con coda di priorità binaria).

Limitazioni: Funziona solo con costi non negativi. Per gestire costi negativi, sono necessari algoritmi come Bellman-Ford.

> Esempi a partire da pagina 130 della dispensa

3.8 7. Domande da Orale e Casi Particolari

1. Relazione tra soluzioni ottime e basi:

> **Domanda:** Può esistere una soluzione ottima al problema del cammino minimo che non sia un albero di copertura?
>
> **Risposta:** Sì, è possibile. Il teorema fondamentale della PL garantisce che *esiste almeno una* soluzione ottima che è un vertice del poliedro (e quindi un albero di copertura per i problemi su rete unimodulari). Tuttavia, se esistono due o più vertici ottimi distinti, allora qualsiasi combinazione convessa di questi vertici è anch'essa una soluzione ottima. Tali combinazioni convesse, non essendo vertici, non sono soluzioni di base. In tal caso, ci sarebbero infinite soluzioni ottime non di base.

2. Degenerazione nel SPT:

> **Domanda:** È vero o falso che il problema dei cammini minimi non ha soluzioni di base degeneri?
>
> **Risposta:** Vero. Nel modello del cammino minimo (in particolare quando si cerca l'albero dei cammini minimi dalla sorgente a tutti gli altri nodi), il flusso su ogni arco dell'albero di base corrisponde al numero di nodi nel sottoalbero radicato in quel punto (che riceve 1 unità di flusso ciascuno). Poiché questo valore è sempre un intero strettamente positivo, nessun arco di base avrà un flusso nullo, quindi non ci possono essere soluzioni di base degeneri. Di conseguenza, la regola anticiclo di Bland non è necessaria per prevenire il *cycling*.

3. Ottimizzazione multiobiettivo:

> **Domanda:** Come si modella un problema di ottimizzazione multiobiettivo (es. minimizzare costo c e tempo l)?
>
> **Risposta:** Non è possibile minimizzare due obiettivi contemporaneamente a meno che non siano proporzionali. La strategia comune è trasformare uno degli obiettivi in un vincolo, introducendo un budget massimo. Ad esempio, per minimizzare il tempo totale di percorrenza ($l^T x$) con un budget massimo sul costo ($c^T x \leq G$):
>
> $$\begin{cases} \min \ell^T x \\ Ex = b \\ 0 \leq x \leq u \\ c^T x \leq G \\ x_{ij} \in \{0,1\} \quad \text{(se si vuole un cammino, non un flusso)} \end{cases}$$
>
> Questo problema **non è più un semplice problema di flusso di costo minimo**, perché il vincolo di budget $c^T x \leq G$ non è un'equazione di bilancio. La matrice dei vincoli non è più totalmente unimodulare, e quindi il **teorema dell'interezza non vale più**. Per trovare un cammino (dove x_{ij} può essere solo 0 o 1), diventa un problema di **Programmazione Lineare Intera (PLI)**, che è generalmente più difficile da risolvere.

4. Problema di Assegnamento come Flusso Massimo (o Accoppiamento Massimo): > **Domanda:** Un'azienda deve assegnare 6 dipendenti a 10 bandi, con vincoli di competenza. Ogni dipendente può fare un solo progetto, e ogni progetto fatto da un solo ingegnere. Come si modella il problema per massimizzare il numero di bandi vinti?
Risposta: Questo problema può essere modellato come un problema di **Flusso Massimo** su un grafo bipartito. > 1. Crea un nodo **sorgente** s e un nodo **pozzo** t. > 2. Crea un nodo per ogni **dipendente** (es. $D_1, \dots, D_6$) e un nodo per ogni **bando** (es. $B_1, \dots, B_{10}$). > 3. Aggiungi archi: > * Da s a ogni D_i, con capacità $u = 1$ (ogni dipendente può essere assegnato al massimo una volta). > * Da ogni D_i a ogni B_j se il dipendente D_i è competente per il bando B_j, con capacità $u = 1$ (un dipendente può lavorare a un bando se competente). > * Da ogni B_j a t, con capacità $u = 1$ (ogni bando può essere assegnato al massimo una volta). > Il flusso massimo da s a t in questa rete corrisponde al numero massimo di assegnazioni possibili, ovvero il numero massimo di bandi che possono essere vinti. Questo è un caso di **accoppiamento massimo** in un grafo bipartito.

5. Flusso Massimo e saturazione degli archi:

> **Domanda:** Può succedere che il flusso massimo non saturi nessun arco?
>
> **Risposta:** No, se il flusso massimo $v > 0$, allora esiste un taglio (N_s, N_t) con capacità v e tutti gli archi diretti di questo taglio devono essere saturi. Altrimenti, se un arco non fosse saturo, il flusso potrebbe essere aumentato, contraddicendo l'ottimalità.

6. Algoritmi Greedy e Ottimalità:

> **Domanda:** È vero che se tutti i costi sono 1000 tranne uno che è 1, quest'ultimo farà sempre parte dell'albero ottimo? E se tutti sono 1 tranne uno a 1000?
>
> **Risposta:** Falso in entrambi i casi. Gli algoritmi per i flussi di costo minimo non sono *greedy* nel senso di scegliere sempre l'arco più economico. La scelta degli archi nell'albero ottimo dipende dalla struttura complessiva della rete e dai bilanci dei nodi, che potrebbero forzare l'utilizzo di archi apparentemente "svantaggiosi" per soddisfare le richieste di flusso.

7. Verifica di ammissibilità e degenerazione:

> **Domanda:** Data una soluzione $x = (6, 1, 0, 3, 7, 5, 0, 0, -2)$, è di base? È ammissibile? È degenere?
>
> **Risposta:**
>
> - **Ammissibilità:** Una soluzione è ammissibile se tutti i flussi sono non-negativi ($x_{ij} \geq 0$) e rispettano i bilanci dei nodi. In questo caso, $x_{54} = -2$, quindi la soluzione **non è ammissibile**.
> - **Di base:** Per essere di base, devono esserci esattamente $m - (n-1)$ zeri (per reti non capacitate) e gli archi con flusso non nullo non devono formare cicli. Se non è ammissibile, non ha senso parlare di "di base" in termini di vertice del poliedro ammissibile.
> - **Degenerazione:** Se non è ammissibile, non si valuta la degenerazione in quanto non è un vertice del poliedro ammissibile. Se fosse stata ammissibile, sarebbe degenere se un arco in T avesse flusso zero.

8. Costruzione di un esempio:

> **Domanda:** Costruire su una maglia un flusso ammissibile, degenere e non ottimo (in una rete capacitata).
> **Risposta:** 1. **Ammissibile:** Assegna flussi che rispettano bilanci e capacità. 2. **Degenerato:** Assicurati che almeno un arco nell'albero di copertura (T) abbia flusso 0 o sia saturo (u_{ij}). Questo rende la soluzione di base degenere. 3. **Non ottimo:** Calcola i potenziali e i costi ridotti. Se una delle condizioni di Bellman è violata (es. $c_{ij}^{\pi} < 0$ per un arco in L, o $c_{ij}^{\pi} > 0$ per un arco in U), allora il flusso non è ottimo.

4 Riassunto Programmazione Non Lineare (PNL)

4.1 Date principali

- **26/11/2024:** Introduzione a PNL, gradiente e Hessiana, minimi/massimi locali.
- **27/11/2024:** Domini ammissibili, convessità, coercività e regolarità.
- **28/11/2024:** Teorema LKKT (Karush-Kuhn-Tucker).
- **02/12/2024:** Condizioni necessarie e sufficienti per ottimo, teoremi.
- **03/12/2024:** Metodo delle restrizioni per l'analisi locale dei punti stazionari candidati.
- **04/12/2024:** Introduzione agli algoritmi iterativi per la PNL, concetto di direzione di salita, passo (step size) e criteri di convergenza.
- **05/12/2024:** Analisi dettagliata degli algoritmi per problemi non vincolati, Metodo del Gradiente, Metodo di Newton e regole di Armijo-Goldstein-Wolfe.
- **09/12/2024:** Algoritmo di Frank-Wolfe per problemi con domini poliedrici e utilizzo di quadprog.
- **11/12/2024:** Algoritmo del Gradiente Proiettato per problemi con domini poliedrici.

4.2 1. Fondamenti di PNL

4.2.1 Problema generale

$$\min f(x) \quad s.t. \quad x \in D \subseteq \mathbb{R}^n$$

- **PL:** D poliedro, f lineare.
- **PNL:** D non poliedro e/o f non lineare.
- Per adattare i metodi di minimizzazione alla massimizzazione, si può massimizzare $f(x)$ minimizzando $-f(x)$.

4.2.2 Derivate e gradiente

La derivata parziale di una funzione $f : \mathbb{R}^n \to \mathbb{R}$ rispetto a x_i nel punto $\bar{x}$ è definita come:

$$\frac{\partial f}{\partial x_i}(\bar{x}) = \lim_{t \to 0} \frac{f(\bar{x} + te_i) - f(\bar{x})}{t}$$

dove e_i è il vettore unitario nella direzione dell'asse i. Questa può essere interpretata come la derivata di una funzione di una variabile $g(t) = f(\bar{x} + te_i)$ calcolata in $t = 0$.

Il gradiente di $f(x)$ è un vettore che raccoglie tutte le derivate parziali:

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right) \in \mathbb{R}^n$$

4.2.3 Hessiana

La matrice Hessiana $H_f(x)$ è una matrice quadrata $n \times n$ contenente le derivate seconde di una funzione f:

$$H_f(x) = \left[\frac{\partial^2 f}{\partial x_i \partial x_j}\right]_{n \times n}$$

- Se H_f è definita positiva in un punto $\bar{x}$, allora $\bar{x}$ è un minimo locale.
- Se H_f è definita negativa in un punto $\bar{x}$, allora $\bar{x}$ è un massimo locale.
- Se H_f è indefinita in un punto $\bar{x}$, allora $\bar{x}$ è un punto sella.

4.2.4 Funzioni Quadratiche

Una funzione quadratica può essere espressa come $f(x) = \frac{1}{2}x^T Q x + c^T x$ o $f(x) = x^T Q x + c^T x$.

- Per $f(x) = \frac{1}{2}x^T Q x + c^T x$, la sua Hessiana è $H_f = Q$.
- Per $f(x) = x^T Q x + c^T x$, la sua Hessiana è $H_f = 2Q$. La matrice Q privilegiata per una funzione quadratica è quella simmetrica, che coincide con l'Hessiana (o è la metà dell'Hessiana a seconda della definizione). C'è una corrispondenza biunivoca tra un polinomio di secondo grado e la sua matrice Hessiana simmetrica.

Esempio

$$f(x) = \underbrace{3x_1^2 + 5x_1x_2 - x_2^2}_{2^\circ\ grado} + \underbrace{4x_1}_{\substack{1^\circ\ grado \\ c=(4,0)}} \qquad n = 2$$

Prima provaiamo il procedimento inverso, data Q devo risalire alla funzione $f(x)$

$$Q = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 + 2x_2 \\ 2x_1 + 3x_2 \end{pmatrix}$$

$$(x_1, x_2) \begin{pmatrix} x_1 + 2x_2 \\ 2x_1 + 3x_2 \end{pmatrix} = x_1^2 + 2x_1x_2 + 2x_1x_2 + 3x_2^2$$

Poniamo caso ai coefficienti adesso, notare che sono gli stessi presenti in Q, $x_1 = 1$, $x_2 = 3$, il coefficiente di x_1x_2 è la somma della controdiagonale, $2 + 2 = 4$. Perciò trovare Q data una funzione quadratica è immediato, attenzione però, la corrispondenza tra matrici e funzioni quadratiche **non è biunivoca**

- Per una matrice Q c'è un unica funzione quadratica
- Per una quadratica le matrici sono infinite, ma **una sola** è privilegiata, quella simmetrica che diventa poi l'Hessiana

Proseguiamo quindi con la funzione iniziale, nella forma $f(x) = \frac{1}{2}x^T(Qx) + c^T x$

$$\nabla f = (6x_1 + 5x_2, 5x_1 - 2x_2)$$

$$Hf = Q = \begin{pmatrix} 6 & 5 \\ 5 & -2 \end{pmatrix}$$

Sappiamo $\nabla f = Qx + c \Rightarrow Hf = Q$, ma nel nostro esempio non è così… questo perchè **abbiamo definito(a livello software)** la funzione come $f(x) = \frac{1}{2}x^T(Qx) + c^T x$, notare il $\frac{1}{2}$

$f(x) = \frac{1}{2}x^T(Qx) + c^Tx$	$f(x) = x^T(Qx) + c^Tx$
$Q = Hf$	$Hf = 2Q$

4.3 2. Teoremi fondamentali

4.3.1 Teorema di Fermat per l'Analisi II

Se $\bar{x}$ è un minimo locale di f, allora $\nabla f(\bar{x}) = 0$. Questa è una *condizione necessaria* per l'ottimalità, ma non sufficiente. Un punto in cui il gradiente è nullo è chiamato punto stazionario e può essere un minimo locale, un massimo locale o un punto sella. Trovare i punti che annullano il gradiente implica risolvere un sistema di n equazioni in n incognite. Per funzioni quadratiche, questo sistema è lineare.

Esempio: Per $f(x_1, x_2) = x_1^2 + x_1x_2 - x_2^2$, il gradiente è $\nabla f = (2x_1 + x_2, x_1 - 2x_2)$. Annullandolo si ottiene il sistema lineare:

$$\begin{cases} 2x_1 + x_2 = 0 \\ x_1 - 2x_2 = 0 \end{cases}$$

La soluzione è $(0, 0)$. Tuttavia, $f(0, 0) = 0$, ma lungo la direzione $(0, x_2)$, $f(0, x_2) = -x_2^2$, che tende a $-\infty$. Questo dimostra che un punto stazionario non è necessariamente un minimo.

4.3.2 Condizioni sulla Hessiana

La classificazione di un punto stazionario $\bar{x}$ può essere determinata analizzando la Hessiana $H_f(\bar{x})$:

$H_f(\bar{x})$	Interpretazione
Definita positiva (> 0)	minimo locale
Semidefinita positiva (≥ 0)	candidato minimo locale (potrebbe essere un minimo, ma non è garantito)
Definita negativa (< 0)	massimo locale
Semidefinita negativa (≤ 0)	candidato massimo locale (potrebbe essere un massimo, ma non è garantito)
Indefinita	punto sella

Una matrice simmetrica è definita positiva se e solo se tutti i suoi autovalori sono strettamente positivi. È semidefinita positiva se e solo se tutti i suoi autovalori sono maggiori o uguali a zero. Le matrici Hessiane sono sempre simmetriche (per funzioni C^2).

Se il gradiente si annulla e la Hessiana è semidefinita (ma non definita), il punto è un caso critico e richiede ulteriori analisi. Un algoritmo che cerca solo le conditions sufficienti potrebbe "girare a vuoto" in questi casi.

4.4 3. Convessità e Coercività

4.4.1 Convessità

Una funzione f è convessa se per ogni x, y nel suo dominio e ogni $\lambda \in [0, 1]$, vale:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

Geometricamente, ciò significa che il segmento di retta che unisce due punti qualsiasi sul grafico della funzione si trova al di sopra o sul grafico stesso.

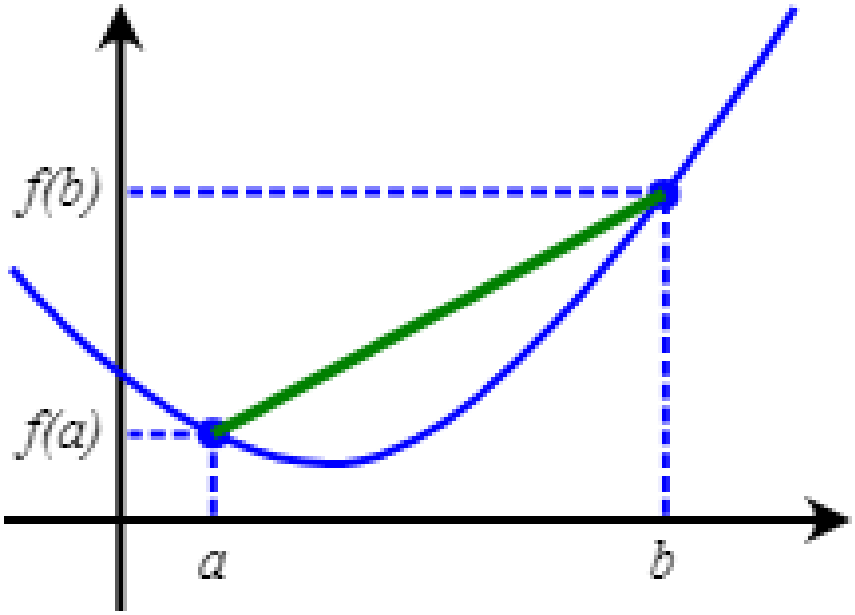

Figura 4.1: *Grafico generico*

Proprietà fondamentali delle funzioni convesse:

- Una funzione è convessa se l'Hessiana è semidefinita positiva $Hf \geq 0$.
- Se $f \in C^2$, f è convessa se e solo se $Hf(x) \geq 0$ per ogni x.
- Se f è convessa e $\nabla f(\bar{x}) = 0$, allora $\bar{x}$ è un minimo globale (mG).
- Una funzione convessa non ha minimi locali che non siano anche minimi globali (ossia, se ha un minimo locale, questo è necessariamente globale).
- Una funzione convessa non ha punti sella.
- Una funzione convessa, se non costante, non ha massimi globali finiti, poiché il gradiente che si annulla implica un minimo globale.

Per le funzioni concave, le disuguaglianze nella definizione si invertono, e la Hessiana deve essere semidefinita negativa.

4.4.2 Coercività

Una funzione f si dice "coerciva" se:

$$\lim_{||x||\to\infty} f(x) = +\infty$$

Questo implica che la funzione tende all'infinito in tutte le direzioni, garantendo l'esistenza di almeno un minimo globale (Teorema di Weierstrass). Una funzione si dice "anticoerciva" se $\lim_{||x||\to\infty} f(x) = -\infty$ (o equivalentemente, $-f$ è coerciva), garantendo l'esistenza di un massimo globale.

- **Esempio 1:** $f(x_1, x_2) = x_1^2 + 3x_2^2 - 5x_1$

La matrice Hessiana H della funzione è:

$$H = \begin{pmatrix} 2 & 0 \\ 0 & 6 \end{pmatrix}$$

Gli autovalori sono $\lambda_1 = 2$ e $\lambda_2 = 6$. Poiché entrambi gli autovalori sono positivi ($\lambda_1 > 0$ e $\lambda_2 > 0$), la matrice H è definita positiva. Questo implica che la funzione quadratica $f(x_1, x_2)$ è **convessa**.

È anche coerciva? **Sì.** Una funzione convessa con una parte quadratica definita positiva è coerciva, poiché i termini quadratici dominano i termini lineari e costanti man mano che $\|x\|$ tende all'infinito, garantendo che $f(x) \to \infty$.

- **Esempio 2:** $f(x_1, x_2) = x_1^2 - x_2^2$

È coerciva? **No.**

Per dimostrarlo, possiamo trovare una direzione lungo la quale la funzione non tende a $+\infty$. Consideriamo la direzione $(0, t)$, che può essere scritta come una semiretta parametrica $x(t) = (0, t)$ per $t \in \mathbb{R}$.

Lungo questa direzione, la funzione diventa:

$$f(x(t)) = f(0, t) = 0^2 - t^2 = -t^2$$

Quando $t \to \infty$ (o $t \to -\infty$), $f(x(t)) = -t^2 \to -\infty$. Poiché esiste una direzione lungo la quale la funzione tende a $-\infty$, la funzione non è coerciva.

Vediamo altri esempi di direzioni per chiarire il comportamento:

- Se $x(t) = t(1, 1)$, allora $f(x(t)) = t^2 - t^2 = 0$, che va a 0.
- Se $x(t) = t(-1, 1)$, allora $f(x(t)) = (-t)^2 - t^2 = t^2 - t^2 = 0$, che va a 0.
- Se $x(t) = t(1, -1)$, allora $f(x(t)) = t^2 - (-t)^2 = t^2 - t^2 = 0$, che va a 0.
- Se $x(t) = t(1, 2)$, allora $f(x(t)) = t^2 - (2t)^2 = t^2 - 4t^2 = -3t^2$, che va a $-\infty$.

4.5 4. Domini ammissibili

Il dominio ammissibile D è l'insieme dei punti $x \in \mathbb{R}^n$ che soddisfano un insieme di vincoli di disuguaglianza e uguaglianza:

$$D = \{x \in \mathbb{R}^n : g_1(x) \leq 0, g_2(x) \leq 0, ..., g_m(x) \leq 0, h_1(x) = 0, ..., h_p(x) = 0\}$$

dove $g_i : \mathbb{R}^n \to \mathbb{R}$ sono i vincoli di disuguaglianza, e $h_j : \mathbb{R}^n \to \mathbb{R}$ sono i vincoli di uguaglianza. Per le applicazioni della PNL, si assume che f, g_i, h_j siano funzioni almeno C^1, preferibilmente C^2. È fondamentale che i domini siano chiusi (che

i vincoli di disuguaglianza includano l'uguaglianza, es. $g_i(x) \leq 0$ e non $g_i(x) < 0$), poiché "il minimo è quasi sempre sul bordo". Se un problema non è chiuso, si può approssimarlo con uno leggermente minore ma chiuso.

Affinché esistano minimi/massimi globali, il dominio D deve essere **chiuso e limitato** (cioè, interamente contenuto in una palla di raggio finito). Questo è garantito dal Teorema di Weierstrass.

Esempi di domini:

1. Cerchio: $g_1(x_1, x_2) = x_1^2 + x_2^2 - 4 \leq 0$.

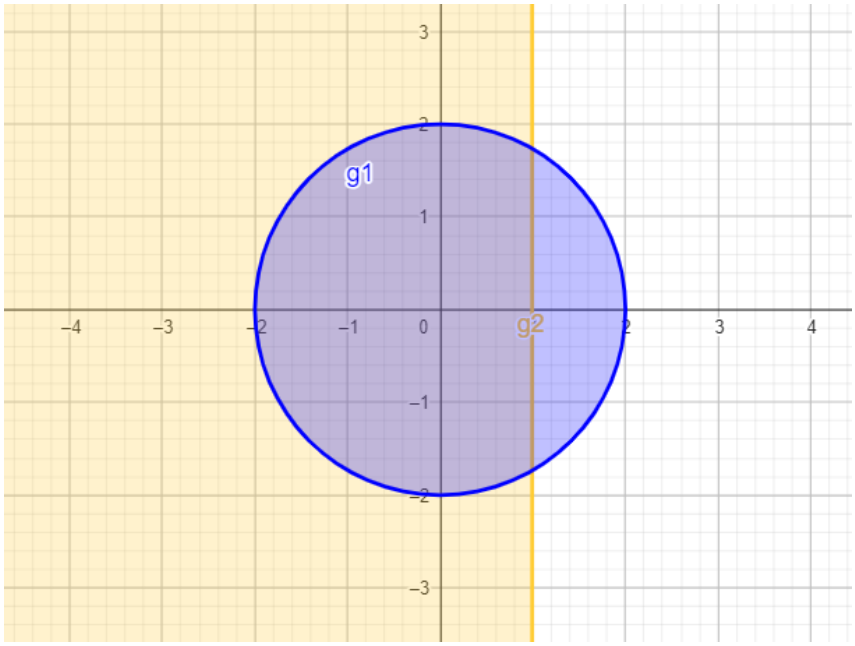

Figura 4.2: *grafico*

2. Parabola: $g_1(x_1, x_2) = x_1 - x_2^2 \leq 0$ (o $x_1 \leq x_2^2$) e $g_2(x_1, x_2) = x_2 - 5 \leq 0$.

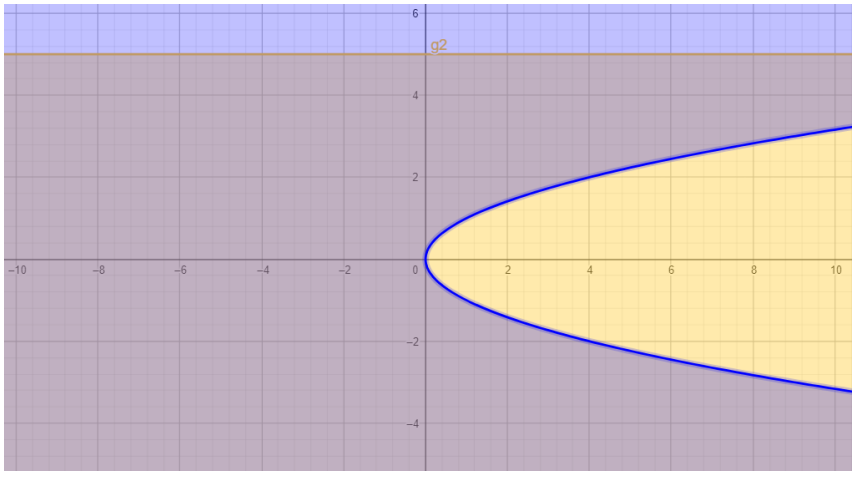

Figura 4.3: *grafico*

3. Dominio convesso: $g_1(x_1, x_2) = -x_1 - x_2^2 \leq 0$ (o $x_1 \geq -x_2^2$) e $g_2(x_1, x_2) = x_1 - 5 \leq 0$.

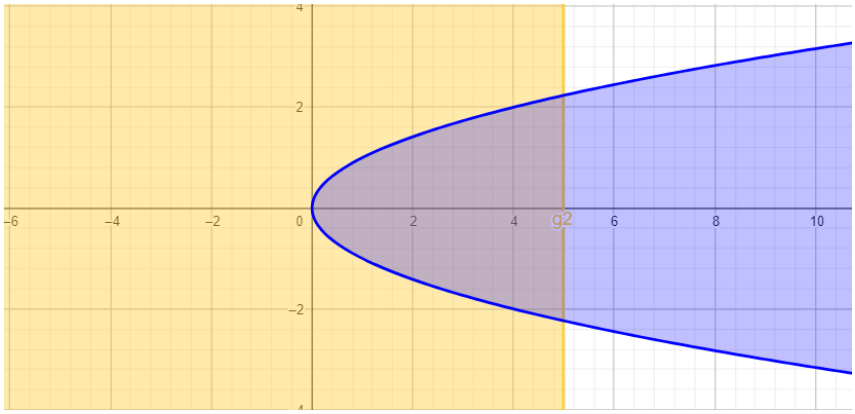

Figura 4.4: *grafico*

4.5.1 Convessità dei Domini

E' fondamentale che il dominio sia convesso e limitato, trovata una direzione di salita se la regione è convessa, appena la soluzione risulterà inammissibile in una certa direzione, lo sarà sempre.
Una condizione sufficiente per la convessità del dominio D è che tutte le funzioni g_i siano convesse e tutte le funzioni h_j siano lineari. (Si ricorda che una funzione g_i è convessa se la sua Hessiana $H_{g_i}(x)$ è semidefinita positiva per ogni x).

4.5.2 Regolarità dei Domini

Un dominio è detto "regolare" se soddisfa certe condizioni che garantiscono la validità dei moltiplicatori di Lagrange. Le principali condizioni di regolarità sono:

1. **D è un poliedro.**
2. **Regolarità secondo Slater:** Esiste un punto $\bar{x} \in D$ tale che $g_i(\bar{x}) < 0$ per tutti i vincoli di disuguaglianza (cioè, $\bar{x}$ è strettamente interno a tutti i vincoli g_i). Questo vale se g_i sono convesse e h_j sono lineari.
3. **Regolarità secondo Mangasarian (LICQ - Linear Independence Constraint Qualification):** I gradienti dei vincoli attivi (quelli per cui $g_i(\bar{x}) = 0$) e dei vincoli di uguaglianza ($h_j(\bar{x}) = 0$) sono linearmente indipendenti nel punto $\bar{x}$ in questione.

Esempio di verifica di regolarità: Consideriamo $D = \{x_1^2 + x_2^2 \leq 1,\ x_1^2 - x_2 \leq 0\}$.

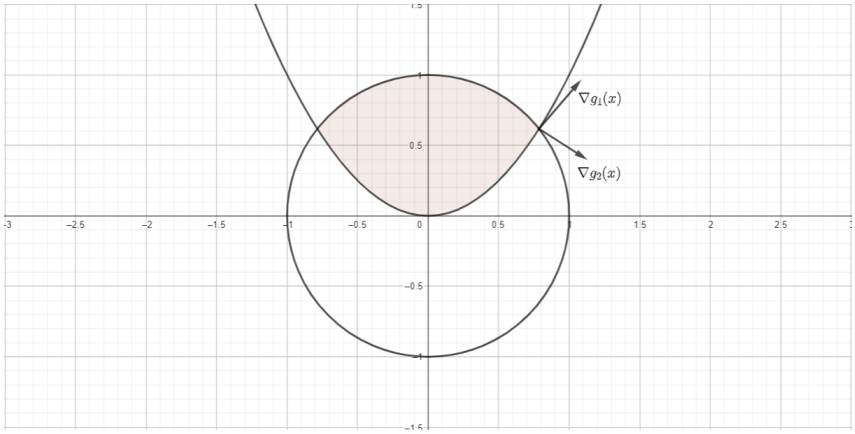

Figura 4.5: *D è l'intersezione*

- Non è un poliedro.
- Rispetta la condizione di Slater, con g_1 e g_2 convesse e l'esistenza di punti interni al dominio.
- Rispetta la condizione di Mangasarian:
  - I gradienti dei vincoli sono $\nabla g_1 = (2x_1, 2x_2)$ e $\nabla g_2 = (2x_1, -1)$.
  - Nei punti di intersezione $(x_1, x_2) = \left(\sqrt{\frac{\sqrt{5}-1}{2}}, \frac{\sqrt{5}-1}{2}\right)$, i gradienti sono linearmente indipendenti (il loro determinante è non nullo). In conclusione: D è un dominio regolare.

4.6 5. Teorema LKKT (Karush-Kuhn-Tucker)

Il Teorema LKKT fornisce le *condizioni necessarie* di ottimalità per problemi di PNL con vincoli. Se $\bar{x} \in D$ è un minimo locale (o massimo locale), e il dominio D è regolare in $\bar{x}$, allora esistono moltiplicatori $\bar{\lambda} \in \mathbb{R}^m$ (per i vincoli di disuguaglianza) e $\bar{\mu} \in \mathbb{R}^p$ (per i vincoli di uguaglianza) tali che:

$$\begin{cases} \nabla f(\bar{x}) + \sum_{i=1}^{m} \bar{\lambda}_i \nabla g_i(\bar{x}) + \sum_{j=1}^{p} \bar{\mu}_j \nabla h_j(\bar{x}) = 0 \quad (\text{stazionarità della Lagrangiana}) \\ < \bar{\lambda},\ g(\bar{x}) >= 0 \quad (\text{condizioni di complementarità}) \\ h(\bar{x}) = 0 \quad (\text{ammissibilità sui vincoli di uguaglianza}) \\ \bar{\lambda}_i \geq 0 \quad (\text{condizione di segno per i moltiplicatori di disuguaglianza}) \end{cases}$$

Questo è il *sistema LKKT*. I λ e μ sono detti *moltiplicatori di Lagrange*.

- La prima equazione è una condizione di stazionarità del Lagrangiano rispetto a x. È un sistema di n equazioni.
- Le condizioni di complementarità ($\bar{\lambda}_i g_i(\bar{x}) = 0$) implicano che se un vincolo $g_i(\bar{x}) < 0$ (non attivo), allora il suo moltiplicatore $\bar{\lambda}_i$ deve essere 0. Se $\bar{\lambda}_i > 0$, allora il vincolo $g_i(\bar{x})$ deve essere attivo (ovvero $g_i(\bar{x}) = 0$).
- Le condizioni $h(\bar{x}) = 0$ assicurano che la soluzione sia ammissibile rispetto ai vincoli di uguaglianza.
- Per un **minimo locale**, i moltiplicatori λ devono essere ≥ 0.
- Per un **massimo locale**, i moltiplicatori λ devono essere ≤ 0.

Il sistema LKKT ha $n + m + p$ equazioni in $n + m + p$ variabili (x, λ, μ).

Interpretazione geometrica dei moltiplicatori di Lagrange: I moltiplicatori possono essere visti come le "forze" o "prezzi ombra" associati ai vincoli attivi che "spingono" la soluzione sul bordo del dominio ammissibile.

Nota dal professore: All'esame, per non rendere il compito un mero esercizio di algebra, il professore solitamente fornisce le coordinate dei punti candidati $\bar{x}$, lasciando allo studente il compito di verificare le condizioni e classificare il punto. Se i moltiplicatori λ sono tutti nulli, il punto potrebbe essere un minimo, un massimo o una sella. Se i segni dei moltiplicatori λ_i sono discordi (alcuni > 0 e altri < 0), il punto è una sella.

> Esempi da pagina 154 della dispensa

4.7 6. Condizioni di ottimalità

4.7.1 Teorema 2 - Condizione sufficiente per minimo globale (mG)

Sia D regolare e $f, g_i, h_j \in C^1$. Se f, g_i per ogni $i = 1, \dots, m$ sono convesse e le h_j per ogni $j = 1, \dots, p$ sono lineari. Se $\bar{x}$ soddisfa le condizioni LKKT con $(\bar{\lambda}, \bar{\mu}) \in \mathbb{R}^{m+p}$ e inoltre $\bar{\lambda} \geq 0$, allora $\bar{x}$ è un minimo globale. Questi sono definiti "problemi convessi di PNL". Il dominio D è convesso sotto queste ipotesi.

4.7.2 Teorema 2-bis - Condizione sufficiente per massimo globale (MG)

Sia D regolare e $f, g_i, h_j \in C^1$. Sia f concava, g_i convesse e h_j lineari. Se $\bar{x}$ soddisfa le condizioni LKKT con $(\bar{\lambda}, \bar{\mu}) \in \mathbb{R}^{m+p}$ e inoltre $\bar{\lambda} \leq 0$, allora $\bar{x}$ è un massimo globale. Questi sono definiti "problemi concavi di PNL".

4.7.3 Teorema 3 - Proprietà per Poliedri limitati

Sia D un poliedro limitato e $f \in C^1$.

1. Se f è convessa, allora il minimo globale (mG) si trova su uno dei vertici di D.
2. Se f è concava, allora il massimo globale (MG) si trova su uno dei vertici di D. Il Teorema 3 è un caso particolare del Teorema 2, applicato ai poliedri.

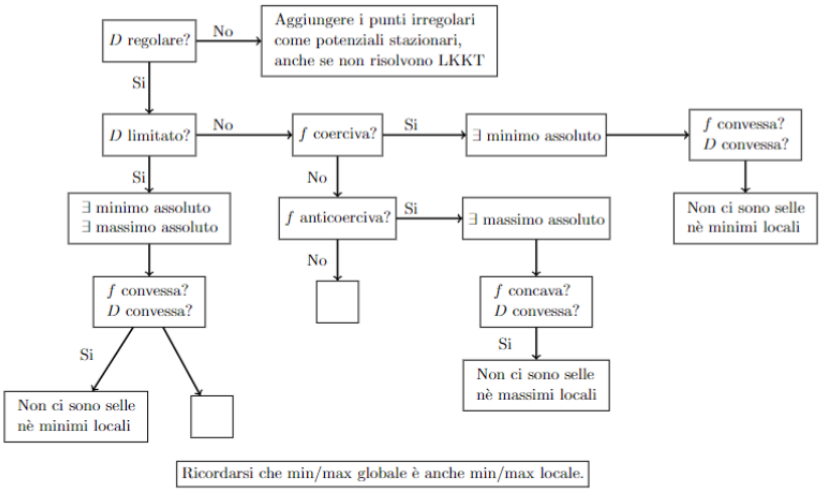

Figura 4.6: schema concettuale per la classificazione

> Esempi da pagina 156 della dispensa

4.8 7. Metodo delle restrizioni

Il metodo delle restrizioni è una tecnica per classificare un punto stazionario (trovato ad esempio tramite LKKT) come minimo locale/massimo locale o sella, specialmente quando le condizioni sulla Hessiana non sono conclusive. L'idea è di analizzare il comportamento della funzione obiettivo lungo le "restrizioni" imposte dai vincoli attivi nel punto in questione.
Principio: Se $\bar{x}$ è minimo locale allora $\bar{x}$ è minimo locale su qualunque "restrizione" (e viceversa). Questo principio è utile per identificare punti sella: se un punto candidato non è un minimo (o massimo) su una determinata restrizione, allora è una sella. Geometricamente, si può visualizzare la funzione ristretta a una curva (ad esempio, il bordo del dominio) e studiare il comportamento della funzione di una variabile risultante.

Esempio: Per $f(x_1, x_2) = (x_1 - 1)^2 + x_2^2$ con $g_1(x_1, x_2) = x_1 - x_2^2 \leq 0$. Si trovano i punti stazionari tramite LKKT. Supponiamo di avere il punto $(0, 0)$ con moltiplicatore $\lambda_1 = 2$. Il dominio è $D = \{(x_1, x_2) : x_1 \leq x_2^2\}$. Il punto $(0, 0)$ è sulla frontiera, e il vincolo g_1 è attivo in $(0, 0)$.

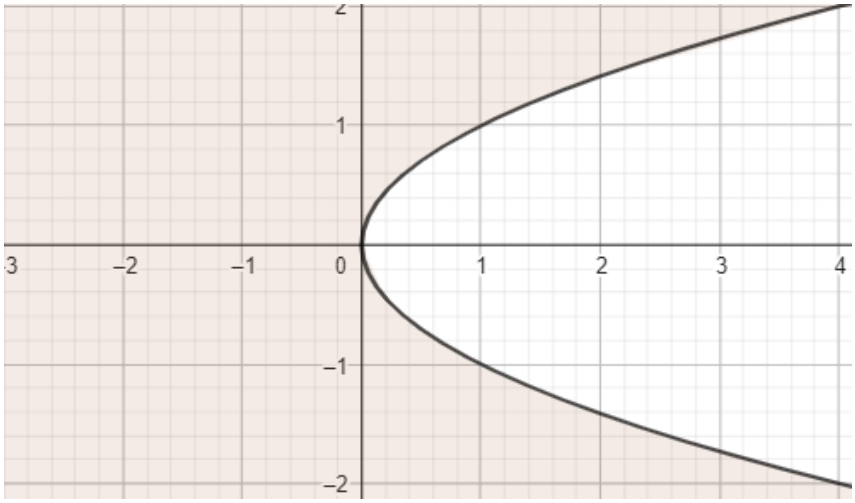

Figura 4.7: *grafico*

Consideriamo la restrizione $x_1 = x_2^2$. Sostituendo nella funzione obiettivo si ottiene: $\phi(x_2) = (x_2^2 - 1)^2 + x_2^2 = x_2^4 - 2x_2^2 + 1 + x_2^2 = x_2^4 - x_2^2 + 1$. Derivando e annullando per trovare i punti stazionari lungo la restrizione: $\phi'(x_2) = 4x_2^3 - 2x_2 = 2x_2(2x_2^2 - 1) = 0$. Le soluzioni sono $x_2 = 0$, $x_2 = \frac{\sqrt{2}}{2}$, $x_2 = -\frac{\sqrt{2}}{2}$. Se $x_2 = 0$, allora $x_1 = 0$, quindi il punto $(0, 0)$. $\phi''(x_2) = 12x_2^2 - 2$. In $x_2 = 0$, $\phi''(0) = -2 < 0$, il che indica un massimo locale per $\phi(t)$.

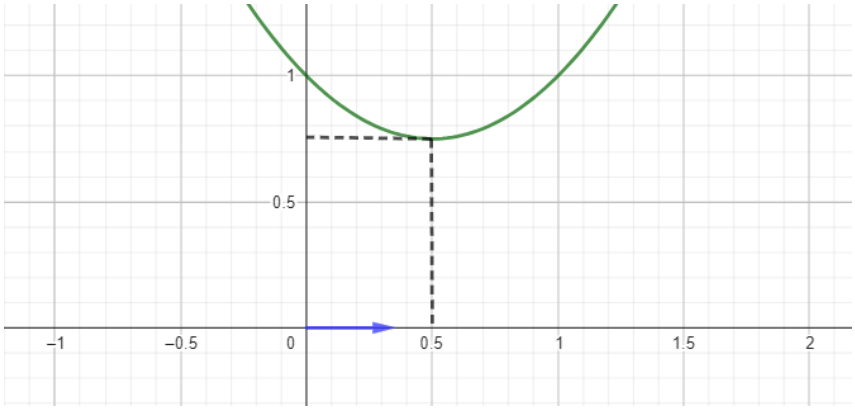

Figura 4.8: $\phi(t)$

Poiché $(0, 0)$ è un massimo locale sulla restrizione $x_1 = x_2^2$, e avevamo un moltiplicatore $\lambda_1 = 2 > 0$ che indicava un possibile minimo, concludiamo che $(0, 0)$ è un punto sella.

4.9 8. Algoritmi di PNL

Gli algoritmi risolutivi per la PNL (spesso per problemi non vincolati o con vincoli semplici) si basano su un processo iterativo:

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

dove:

- $x^k \in \mathbb{R}^n$ è il punto corrente all'iterazione k.
- $d^k \in \mathbb{R}^n$ è la direzione di ricerca (tipicamente di salita per la massimizzazione, di discesa per la minimizzazione).
- $t_k \in \mathbb{R}, t_k \geq 0$, è il passo o "step size" lungo la direzione d^k.

L'obiettivo è generare una successione $\{x^k\}$ tale che $f(x^{k+1}) > f(x^k)$ per la massimizzazione (o $f(x^{k+1}) < f(x^k)$ per la minimizzazione). Una direzione d^k è una **direzione di salita** per f in x^k se $\nabla f(x^k) \cdot d^k > 0$. Questo perché la derivata direzionale $\varphi'(0) = \nabla f(x^k) \cdot d^k$, dove $\varphi(t) = f(x^k + td^k)$, indica la pendenza iniziale lungo d^k. Se è positiva, la funzione aumenta inizialmente in quella direzione.

4.9.1 Criteri di stop

Gli algoritmi iterativi in PNL raramente garantiscono il raggiungimento del massimo globale in un numero finito di passi (a meno di casi speciali, come funzioni quadratiche). Si mira a trovare un punto stazionario e si utilizzano criteri di stop pratici:

1. **Numero massimo di iterazioni:** L'algoritmo si ferma dopo un certo numero predefinito di passi.
2. **Norma del gradiente sufficientemente piccola:** $||\nabla f(x^k)|| \leq \epsilon$, dove ϵ è una piccola tolleranza. Indica che il punto è "quasi" stazionario.
3. **Miglioramento della funzione obiettivo trascurabile:** $f(x^k) - f(x^{k-1}) \leq \epsilon$. Se la funzione non migliora significativamente tra due iterazioni consecutive.
4. **Tempo massimo di esecuzione:** L'algoritmo si ferma dopo un certo periodo di tempo.

4.9.2 Teorema di convergenza comune

Per molti algoritmi di PNL, vale il seguente teorema:

- **Ipotesi:** I *level sets* $lev_{\geq k} f = \{x \in \mathbb{R}^n : f(x) \geq k\}$ (per la massimizzazione) sono compatti (chiusi e limitati) per ogni k. Questa ipotesi assicura l'esistenza di un massimo.
- **Tesi:** La successione $\{x^k\} \subseteq \mathbb{R}^n$ generata dall'algoritmo ha punti di accumulazione, e ognuno di questi punti di accumulazione $\bar{x}$ è stazionario ($\nabla f(\bar{x}) = 0$). Se la funzione f è concava (per massimizzazione) o convessa (per minimizzazione), il punto stazionario trovato è un ottimo globale.

4.9.3 Metodi principali per funzioni libere (non vincolate)

4.9.3.1 Algoritmo 1 - metodo del gradiente a passo costante

$$\begin{cases} d^k = \nabla f(x^k) \\ t^k = \eta \in \mathbb{R} \end{cases}$$

$x^{k+1} = x^k + \eta \nabla f(x^k)$

4.9.3.1.1 Teorema convergenza $\{x^k\}$ ha almeno 1 punto di accumulazione e ogni punto di accumulazione $\bar{x}$ è tale che $\nabla f(\bar{x}) = 0 \ \forall x^0_{pt.\ iniziale}$

- $0 < \eta \leq \frac{2}{L}$ dove L è tale che $\left|\frac{\delta f}{\delta x_i \delta x_j} \leq L\right|$ rappresenta una stima superiore delle derivate seconde

4.9.3.2 Algoritmo 2 - metodo del gradiente a passo "ideale"

$$\begin{cases} d^k = \nabla f(x^k) \\ t^k \in \underset{t \geq 0}{argmax} \underbrace{f(x^k + td^k)}_{\phi(t)} \end{cases}$$

$x^{k+1} = x^k + t^k \nabla f(x^k)$

4.9.3.2.1 Teorema convergenza Come sopra

$\{x^k\}$ ha almeno 1 punto di accumulazione e ogni punto di accumulazione $\bar{x}$ è tale che $\nabla f(\bar{x}) = 0 \ \forall x^0_{pt.\ iniziale}$

> Mi muovo nella direzione di salita e mi fermo al massimo lungo quella direzione
>
> In una forma quadratica la $\phi(t)$ rimane quadratica, esempio: da una parabola trovo un'altra parabola, e trovarne il massimo è banale

4.9.3.3 Algoritmo 3 - metodo di Newton

$$\begin{cases} d^k = Hf(x^k)^{-1} \ \nabla f(x^k) \\ t^k = 1 \end{cases}$$

$x^{k+1} = x^k + Hf(x^k)^{-1} \ \nabla f(x^k)$

> Non ho la sciurezza che la prima condizione sia di salita.

4.9.3.3.1 Teorema convergenza $\{x^k\}$ ha almeno 1 punto di accumulazione e ogni punto di accumulazione $\bar{x}$ è tale che $\nabla f(\bar{x}) = 0 \ \forall x^0$ vicino al punto di massimo.

> Si suppone Hf invertibile, farla ad ogni passo può essere pesante, inoltre è limitante anche il punto di partenza, se fosse lontano dal massimo potrebbe *impazzire*. Questo algoritmo nella pratica è una *scheggia*.

4.9.3.4 4 Regole di Armijo-Goldstein-Wolfe

$$t_k : \begin{cases} (1)\phi(t_k) \geq \phi(0) + \alpha\phi'(0)t_k \\ (2)\phi'(t_k) \leq \beta\phi'(0) \end{cases}$$

$$d^k = \nabla f(x^k) \quad 0 < \alpha < \beta < 1$$

- $(1)\alpha > 0, \phi'(0) > 0 \rightarrow t$ non può essere molto grande
- $(2) se \ t_k \rightarrow 0$ per la conclusione prima raggiunta, $\phi'(t_k) \rightarrow \phi'(0)$, quindi t_k non può essere troppo piccolo
- $\exists(A, B)$ in cui le due regole valgono contemporaneamente

> Mi muovo da $f(x^k)\ verso\ f(x^{k+1})$ incrementando il primo di $\alpha\phi'(0)t_k$

4.9.3.4.1 Metodo *backtracing* Si parte con un valore $x = \bar{x} > 0$ e, finché la condizione (a) di Armijo–Goldstein–Wolfe non è soddisfatta, si moltiplica x per un fattore $\gamma \in (0, 1)$. Questa procedura garantisce che il punto trovato soddisfi la condizione di discesa per f, ma assicura anche che il passo non sia troppo piccolo.

1. Fissa $\alpha, \gamma \in (0, 1)$ e $\bar{x} > 0$.
2. Calcola il più piccolo numero naturale m tale che

$$f(\gamma^m \bar{x}) \leq f(0) + \alpha f'(0)\gamma^m \bar{x}.$$

3. Poni $x := \gamma^m \bar{x}$.

> Esempi sui metodi da pagina 166

4.9.4 Algoritmo di Frank-Wolfe (per domini poliedrici limitati)

L'algoritmo di Frank-Wolfe è usato per risolvere problemi di massimizzazione (o minimizzazione) di una funzione $f(x)$ su un dominio poliedrico limitato P.

1. **Passo iniziale:** Scegli un punto $x^k \in P$.
2. **Direzione di ricerca:** Risolvi un problema lineare ausiliario (PL) per trovare la direzione y^k:

$$\max_{x \in P} \nabla f(x^k) \cdot x \quad \rightarrow \quad y^k \quad \text{(vertice ottimo del PL)}$$

   Il punto y^k sarà uno dei vertici del poliedro P. La direzione di ricerca è $d^k = y^k - x^k$. Geometricamente, d^k è la direzione che va dal punto corrente x^k al vertice y^k che massimizza la proiezione del gradiente.

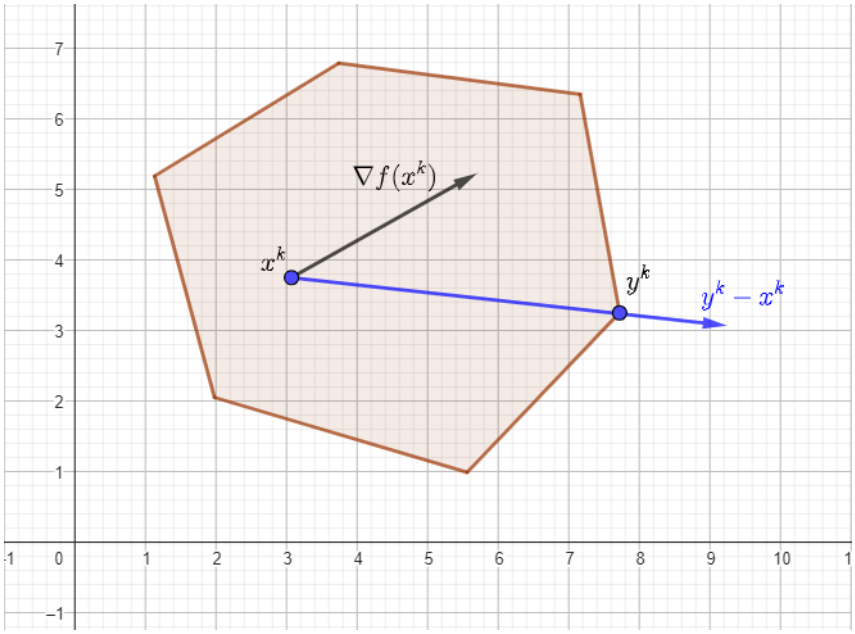

Figura 4.9: graficamente

3. **Calcolo del passo:** Trova $t_k \in [0,1]$:
$$t_k \in \underset{0 \le t \le 1}{argmax} f(x^k + t(y^k - x^k))$$
Questo è un problema di massimizzazione di una funzione di una variabile su un intervallo limitato. Il passo ideale può essere sostituito con le regole AGW, adattandole per $t \in [0,1]$.

4. **Aggiornamento:** $x^{k+1} = x^k + t_k d^k$.

5. **Criterio di stop:** Se x^k è un punto stazionario (ad esempio, se $d^k = 0$ o se il miglioramento $f(x^{k+1}) - f(x^k)$ è trascurabile), ferma l'algoritmo. La successione $\{x^k\}$ ha almeno un punto di accumulazione, e tutti i punti di accumulazione sono soluzioni LKKT. Se f è concava, Frank-Wolfe converge al massimo globale.

Esempio di Frank-Wolfe: $f(x) = 2x_1^2 + 4x_2^2 + 4x_1 - x_2$, MAX. Poliedro P definito da vertici $(0,1), (4,1), (4,3), (-1,5)$. Punto iniziale $x^k = (4, 5/3)$.

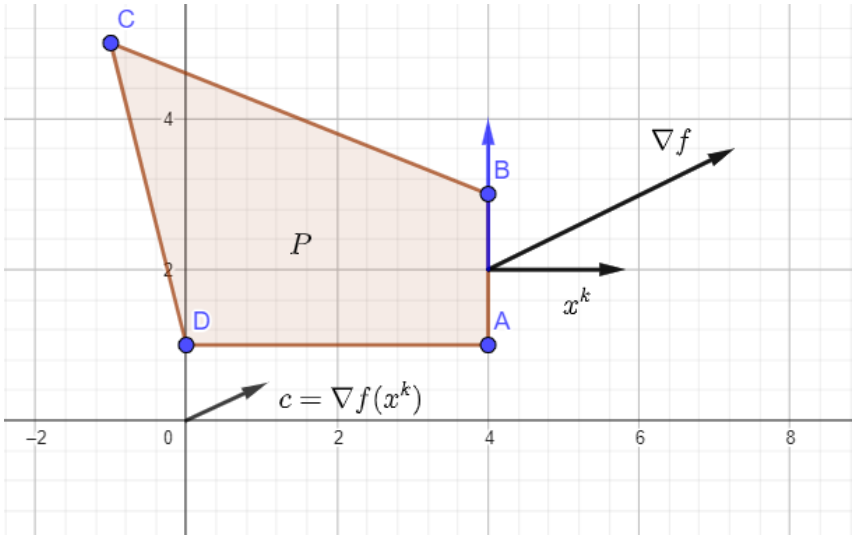

Figura 4.10: *grafico*

1. $\nabla f = (4x_1 + 4, 8x_2 - 1)$.
2. $\nabla f(4, 5/3) = (4(4) + 4, 8(5/3) - 1) = (20, 37/3)$.
3. PL: $\max_{x \in P}(20x_1 + \frac{37}{3}x_2)$. Per trovare il vertice y^k che massimizza questa funzione lineare, si valutano i vertici del poliedro.

   - $(0, 1) \to 20(0) + \frac{37}{3}(1) = \frac{37}{3} \approx 12.33$
   - $(4, 1) \to 20(4) + \frac{37}{3}(1) = 80 + \frac{37}{3} = \frac{277}{3} \approx 92.33$
   - $(4, 3) \to 20(4) + \frac{37}{3}(3) = 80 + 37 = 117$
   - $(-1, 5) \to 20(-1) + \frac{37}{3}(5) = -20 + \frac{185}{3} = \frac{125}{3} \approx 41.67$ Il vertice ottimo è $y^k = (4, 3)$.

4. Direzione di ricerca: $d^k = y^k - x^k = (4, 3) - (4, 5/3) = (0, 4/3)$.
5. Calcolo del passo t_k, usiamo il passo ideale(richiesto all'esame): $\phi(t) = f(x^k + td^k) = f(4, 5/3 + 4/3t) = 2(4)^2 + 4(5/3 + 4/3t)^2 + 4(4) - (5/3 + 4/3t)$.
   $\phi(t) = 32 + 4(\frac{25}{9} + \frac{40}{9}t + \frac{16}{9}t^2) + 16 - \frac{5}{3} - \frac{4}{3}t = \frac{139}{3} + \frac{148}{9}t + \frac{64}{9}t^2$. Questa è una parabola che apre verso l'alto (è convessa, il problema è di massimizzazione).
   Il massimo si troverà a $t = 1$ o $t = 0$ o un punto intermedio se $\phi'(t) = 0$. Per $t \in [0, 1]$, $\phi'(t) = \frac{148}{9} + \frac{128}{9}t$.
   Essendo sempre positiva per $t \in [0, 1]$, il massimo si trova in $t_k = 1$.
6. Aggiornamento: $x^{k+1} = x^k + 1 \cdot d^k = (4, 5/3) + (0, 4/3) = (4, 3)$. Il punto $(4, 3)$ è il nuovo punto. Se si continua l'algoritmo, in $(4, 3)$ il gradiente è $\nabla f(4, 3) = (20, 23)$. Il problema PL darà ancora $(4, 3)$ come y^k. Quindi $d^k = (0, 0)$ e l'algoritmo si ferma. Il punto $(4, 3)$ è una soluzione LKKT. Per trovare il massimo globale, occorre valutare f in tutti i vertici del poliedro.

$$f(0, 1) = 2(0)^2 + 4(1)^2 + 4(0) - 1 = 3$$

$$f(4, 1) = 2(4)^2 + 4(1)^2 + 4(4) - 1 = 32 + 4 + 16 - 1 = 51$$

$$f(4, 3) = 2(4)^2 + 4(3)^2 + 4(4) - 3 = 32 + 36 + 16 - 3 = 81$$

$$f(-1,5) = 2(-1)^2 + 4(5)^2 + 4(-1) - 5 = 2 + 100 - 4 - 5 = 93$$

Il massimo globale è 93, raggiunto in $(-1,5)$. Il punto $(4,3)$ è un massimo locale.

4.9.4.1 quadprog

Per i moltiplicatori $\lambda_1, \lambda_2, \lambda_3, \lambda_4$

Il punto (-1,5) calcolato nel vincolo $\lambda_4, x_2 \geq 1 \rightarrow -x_2 \leq -1 \rightarrow \lambda_4 g_4 = 0 \rightarrow \lambda_4 = 0\ e\ \lambda_3 = 0 \ldots (g_4 = -4)$

Gli unici saranno $\lambda_1\ e\ \lambda_2$, si ottiene un sistemino LKKT semplice 4x4

`quadprog(Q,C,A,b,Aeq,beq,LB,UB)`

$$\begin{cases} \max\ \frac{1}{2} xQx + c^T x \\ Ax \leq b \\ Aeqx = beq \\ LB \leq x \leq UB \end{cases}$$

Nel nostro caso

$$Q = \begin{pmatrix} 4 & 0 \\ 0 & 8 \end{pmatrix}$$

> `Quadprog` è affidabile solo per minimi di convesse e massimi di concave

> Come cerco il minimo senza `quadprog`? In primis dovrei fare il gradiente, trovo il punto che lo annulla, nel nostro caso, (-1, 1/8) che è fuori dal poliedro. Il che ha senso, funzioni convesse nel punto in cui si annulla il gradiente hanno minimo assoluto.

4.9.5 Algoritmo del Gradiente Proiettato (per domini poliedrali limitati)

L'algoritmo del Gradiente Proiettato è utilizzato per problemi di massimo (o minimo) su un poliedro.

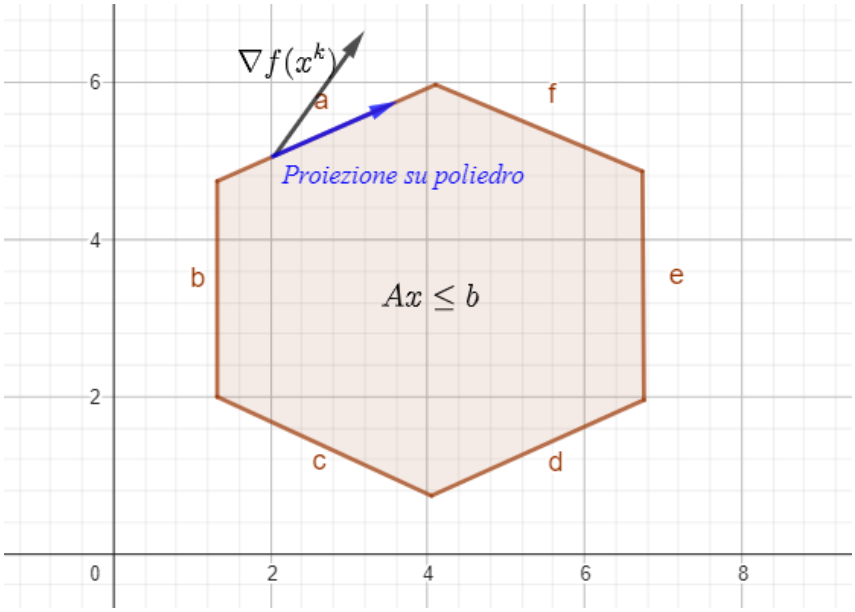

Figura 4.11: *proiezione*

Supponiamo la proiezione sia non nulla, l'angolo è minore di $\frac{\pi}{2}$, è un direzione di crescita.

$$\nabla f(x^k)d^k > 0 \rightarrow d^k \text{ direzione di salita}$$

- $= 0$ nessuna informazione
- < 0 direzione di discesa

Dato x^k prendo gli indici attivi $A(x^k) = \{i \in I : A_i x^k = b_i\}$

Prendo M la sottomatrice di A formata dagli indici attivi $i \in A(x^k)$

4.9.5.1 Matrice di proiezione

$$H = \begin{cases} I & \text{se } x^k \text{ non ha vincoli attivi} \\ \underset{n\times n}{I} - \underset{n\times k}{M^T}(\underset{k\times k}{MM^T})^{-1}\underset{k\times n}{M} & \text{altrimenti} \end{cases}$$

Con *k* vincoli attivi ed *n* varibili.

4.9.5.2 Teorema

$H \cdot y$ è la proiezione di y su M.

$$d^k = H \cdot \nabla f(x^k) \rightarrow \text{gradiente proiettato}$$

4.9.5.3 Massimo spostamento

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

$t_k \in \underset{t \in [0, \hat{t}_k]}{argmax}\ f(x^k + t \underset{H \cdot \nabla f(x^k)}{d^k})$

4.9.5.4 Teorema convergenza

La successione $\{x_k\}$ costruita dall'algoritmo del **gradiente proiettato**:

1. Converge a un **punto stazionario** x^*.
2. Se f è **convessa**, converge al **minimo globale** (niente minimi locali o selle).

Se f è **quadratica e convessa**, la convergenza avviene in un **numero finito di passi**.

4.9.5.5 Passi dell'algoritmo in chiaro

4.9.5.5.1 Passo 1: Punto iniziale Parto con un punto iniziale x^0, con $k = 0$.

4.9.5.5.2 Passo 2: Vincoli attivi Individuo i vincoli attivi ($A_i x^k = b_i$) e costruisco la matrice dei vincoli attivi:

$$M = \begin{bmatrix} A_{i_1} \\ A_{i_2} \\ \vdots \end{bmatrix}$$

4.9.5.5.3 Passo 3: Proiezione Costruisco la matrice di proiezione:

$$H = \begin{cases} I & \text{se } x^k \text{ non ha vincoli attivi} \\ \underset{n \times n}{I} - \underset{n \times k}{M^T} (\underset{k \times k}{M M^T})^{-1} \underset{k \times n}{M} & \text{altrimenti} \end{cases}$$

Calcolo la direzione:

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

4.9.5.5.4 Passo 4: Determinazione dello step

- Se $d_k = 0$: risolvo il caso particolare.
- Altrimenti:
  1. Trovo $\hat{t}_k$.
  2. Calcolo: $x^{k+1} = x^k + t_k d^k$.
  3. Aggiorno $k = k + 1$ e torno al passo 2.

4.9.5.6 Problema di minimo

Se voglio risolvere un **problema di minimo**, pongo:

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

Oltre ad invertire segno al gradiente, l'algoritmo rimane invariato.

4.9.5.7 Riepilogo passaggi (max / min)

1) $\nabla f(x^k)$
2) Matrice M
3) matrice $H \longrightarrow$ se $d^k = 0$ vai a 6)

$$\begin{cases} H = I - M^T(MM^T)^{-1}M \\ d^k = H\nabla f(x^k) \end{cases}$$

4) $\hat{t}_k$
5) $argmax_{t\in[0,\hat{t}_k]}\Psi(t)$ / $argmin_{t\in[0,\hat{t}_k]}\Psi(t)$
6) $\lambda \longrightarrow \leq (\geq) 0$ stop
7) $\lambda_j = \max \lambda_i (> 0)$ /$\lambda_j = \min \lambda_i (< 0)$ si elimina una riga da M e si va a 3)

4.10 9. Esempio di problema PNL

Questo problema riguarda l'ottimizzazione della disposizione di 6 robot su un piano per minimizzare la lunghezza totale dei cavi in fibra ottica che li connettono, rispettando i vincoli di non sovrapposizione delle aree di lavoro.

$$\min \sum_{i<j} d_{ij}, \quad s.t. \quad d_{ij} \geq r_i + r_j \quad \forall i \neq j$$

- **Variabili:** Le coordinate (x_k, y_k) per ogni robot $k = 1, \dots, 6$. In totale, $2 \times 6 = 12$ variabili.
- **Funzione obiettivo:** La somma delle lunghezze dei cavi tra tutti i robot. Questo si traduce nella somma delle distanze euclidee tra ogni coppia di robot i e j: $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

$$\min \sum_{i<j} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

- **Vincoli:** Le aree di lavoro dei robot non devono sovrapporsi. Questo significa che la distanza tra due robot i e j deve essere maggiore o uguale alla somma dei loro raggi d'azione ($r_i + r_j$).

$$\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \geq r_i + r_j \quad \forall i \neq j$$

Per rendere le funzioni differenziabili (requisito per molti algoritmi di PNL), si possono elevare al quadrato entrambi i lati dei vincoli (poiché entrambi i lati sono positivi):

$$(x_i - x_j)^2 + (y_i - y_j)^2 \geq (r_i + r_j)^2$$

Si noti che la funzione obiettivo (somma di radici quadrate) non è differenziabile quando la distanza è zero. Tuttavia, i vincoli di non sovrapposizione impediscono che le distanze diventino nulle.

4.11 Domande d'esame comuni

> **Perché la Hessiana definita positiva implica minimo locale?** Perché una Hessiana definita positiva in un punto significa che la funzione è localmente convessa in quel punto. Per funzioni differenziabili, la convessità locale in un intorno di un punto stazionario (dove il gradiente è nullo) implica che quel punto è un minimo locale.

> **Qual è l'interpretazione geometrica dei moltiplicatori di Lagrange?** I moltiplicatori di Lagrange (λ, μ) possono essere interpretati come le "forze" o "prezzi ombra" associati ai vincoli attivi che mantengono la soluzione sul bordo del dominio ammissibile.

> **Differenza tra condizioni necessarie e sufficienti in PNL?** Le **condizioni necessarie** devono valere in un punto di ottimo, ma il loro rispetto non garantisce che il punto sia ottimo. Ad esempio, un gradiente nullo è necessario per un minimo locale, ma il punto potrebbe anche essere un massimo locale o una sella. Le **condizioni sufficienti**, se soddisfatte, garantiscono che il punto sia un ottimo. Ad esempio, un gradiente nullo e una Hessiana definita positiva garantiscono che il punto sia un minimo locale.

4.12 Tabella comparativa degli algoritmi

Questa tabella riassume le caratteristiche principali di diversi algoritmi, sia per la Programmazione Lineare (PL) che per la Programmazione Non Lineare (PNL).

Algoritmo	Punto Iniziale	Criterio di Stop	Convergenza	Numero di Passi
Simplesso (PL)	Soluzione di base ammissibile	Primale: duale ammissibile / Duale: primale ammissibile	Soluzione ottima (globale)	Sicuramente finito, nel caso peggiore cresce esponenzialmente in base al numero di vincoli e variabili
Simplesso Reti	Soluzione di base ammissibile	Complemen. ammissibile	Soluzione ottima	Sicuramente finito
Piani di Taglio	Soluzione ottima del problema continuo	Ottimo di P_{new} ha soluzioni intere	Soluzione ottima	

Algoritmo	Punto Iniziale	Criterio di Stop	Convergenza	Numero di Passi
Branch & Bound	Soluzione ottima del rilassato continuo	Migliore soluzione ammissibile per la PL dopo aver visitato tutto l'albero	Soluzione ottima	Nel caso peggiore 2^n (n=n° archi TSP, n=n° variabili zaino)
Dijkstra	Nodo sorgente	$N = \emptyset$ (visita tutti i nodi)	Albero cammini minimi	n
Ford-Fulkerson	Soluzione $\bar{x}$ ammissibile con valore ass.	Non ci sono altri cammini ammissibili che raggiungono il nodo destinazione	Flusso max	Numero finito, a patto che le capacità abbiano valori razionali
Gradiente Libero (tutti i tipi)				
Frank-Wolfe	x^k	Il punto soddisfa il sistema LKKT	Max o min	Sempre finito (?)
Gradiente Proiettato	x^k	Min: $\lambda \geq 0$ / Max: $\lambda \leq 0$	Max o min	Sempre finito