COMPLEMENTI DI RICERCA OPERATIVA

Prof. Marco Trubian
6 CFU

Luca Cappelletti

Lecture Notes
Year 2017/2018

Magistrale Informatica
Università di Milano
Italy
27 novembre 2018

Indice

Parte I

Programmazione non lineare

1

Alcune definizioni base

1.1 La convessità

Definizione 1.1.1 (Insieme convesso). Un insieme $X \subset \mathbb{R}^n$ è convesso se comunque presi due punti $\underline{x}, \underline{y} \in X$, allora $\lambda\underline{x} + (1-\lambda)\underline{y} \in X$, per ogni $\lambda \in [0,1]$, cioè tutti i punti della **combinazione convessa** appartengono all'insieme convesso da cui sono presi i due punti.

La proprietà di convessità è invariante rispetto alle operazioni di moltiplicazione con uno scalare, unione e intersezione con un altro insieme convesso.

Definizione 1.1.2 (Funzione convessa). Una funzione $f : \mathbb{R}^n \to \mathbb{R}$ è convessa se il suo dominio è un insieme convesso $X \subseteq \mathbb{R}^n$ e comunque presi due punti $\underline{x}, \underline{y} \in X$, il valore della funzione nella combinazione convessa risulta minore o uguale alla combinazione convessa dei valori della funzione nei due punti:

$$f(\lambda\underline{x} + (1-\lambda)\underline{y}) \leqslant \lambda f(\underline{x}) + (1-\lambda) f(\underline{y}) \qquad \forall \lambda \in [0,1]$$

La proprietà di convessità è invariante rispetto a moltiplicazione con uno scalare e somma tra funzioni convesse.

Vale inoltre che la funzione max di una o più funzioni convesse e che il luogo dei punti $\underline{x}$ per i quali vale che $f(\underline{x}) \leqslant \alpha$ è convesso.

Definizione 1.1.3 (Problema convesso). Un problema di ottimizzazione con funzione obiettivo e regione ammissibile entrambe convesse viene detto problema convesso.

1.2 Massimi e minimi locali

Definizione 1.2.1 (Minimo globale). Un punto $\underline{x}^* \in X$ è un punto di minimo globale di $f(\underline{x})$ se:

$$f(\underline{x}^*) \leqslant f(\underline{x}) \quad \forall \underline{x} \in X$$

Definizione 1.2.2 (Minimo locale). Un punto $\underline{x}^* \in X$ è un punto di minimo locale di $f(\underline{x})$ se esiste un intorno aperto $I(\underline{x}^*, \epsilon)$ di $\underline{x}^*$ avente raggio $\epsilon > 0$ tale che:

$$f(\underline{x}^*) \leqslant f(\underline{x}) \quad \forall \underline{x} \in X \cap I(\underline{x}^*, \epsilon)$$

2

Ottimizzazione non vincolata

Definizione 2.0.1 (Direzione di discesa). Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$, un vettore $\underline{\boldsymbol{d}} \in \mathbb{R}^n$ si dice **direzione di discesa** per f in $\underline{\boldsymbol{x}}$ se il valore della funzione del punto ottenuto seguendo la direzione per uno step λ risulta minore del valore iniziale:

$$\exists \lambda > 0 : f(\underline{\boldsymbol{x}} + \lambda \underline{\boldsymbol{d}}) < f\left(\underline{\boldsymbol{x}}\right)$$

Definizione 2.0.2 (Derivata direzionale). Sia data una funzione $f : \mathbb{R}^n \to \mathbb{R}$, un vettore $\underline{\boldsymbol{d}} \in \mathbb{R}^n$ e un punto dove f è definita. Se esiste il limite:

$$\lim_{\lambda \to 0^+} \frac{f(\underline{\boldsymbol{x}} + \lambda \underline{\boldsymbol{d}}) - f\left(\underline{\boldsymbol{x}}\right)}{\lambda}$$

allora tale limite prende il nome di **derivata direzionale** della funzione f nel punto $\underline{\boldsymbol{x}}$ lungo la direzione $\underline{\boldsymbol{d}}$

2.1 Condizioni necessarie di ottimalità del primo ordine

Teorema 2.1.1 (Condizioni necessarie di ottimalità del primo ordine). Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$, derivabile in $\underline{\boldsymbol{x}}^* \in \mathbb{R}^n$, condizione necessaria affinché il punto $\underline{\boldsymbol{x}}^*$ sia un minimo locale per f è che il gradiente della funzione calcolato in $\underline{\boldsymbol{x}}^*$ sia nullo.

$$\nabla f\left(\underline{\boldsymbol{x}}^*\right) = 0$$

2.2 Condizioni necessarie di ottimalità del secondo ordine

Teorema 2.2.1 (Condizioni necessarie di ottimalità del secondo ordine). Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$ di classe $\underline{\boldsymbol{C}}^2\left(\underline{\boldsymbol{x}}^*\right)$, condizione necessarie affinché il punto $\underline{\boldsymbol{x}}^*$ sia un minimo locale per f è che il gradiente della funzione calcolato in $\underline{\boldsymbol{x}}^*$ sia nullo e che valga la relazione seguente:

$$\underline{\boldsymbol{d}}^T \boldsymbol{H}\left(\underline{\boldsymbol{x}}^*\right) \underline{\boldsymbol{d}} \geq 0 \quad \forall \underline{\boldsymbol{d}} \in \mathbb{R}^n$$

Cioè l'hessiana è definita come **semi-positiva**.

2.3 Condizioni necessarie di ottimalità in senso stretto del secondo ordine

Teorema 2.3.1 (Condizioni necessarie di ottimalità del secondo ordine). Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$ di classe $\underline{\boldsymbol{C}}^2\left(\underline{\boldsymbol{x}}^*\right)$, condizione necessarie affinché il punto $\underline{\boldsymbol{x}}^*$ sia un minimo locale in **senso stretto** per f è che il gradiente della funzione calcolato in $\underline{\boldsymbol{x}}^*$ sia nullo e che valga la relazione seguente:

$$\underline{\boldsymbol{d}}^T \boldsymbol{H}\left(\underline{\boldsymbol{x}}^*\right) \underline{\boldsymbol{d}} > 0 \quad \forall \underline{\boldsymbol{d}} \in \mathbb{R}^n$$

Cioè la matrice hessiana $\boldsymbol{H}$ è **definita positiva**.

3

Programmazione quadratica

Nella programmazione quadratica si approssima f con il seguente modello quadratico, dove $\boldsymbol{Q}$ è la matrice hessiana:

$$\min f\left(\underline{\boldsymbol{x}}\right) = \frac{1}{2}\underline{\boldsymbol{x}}^T \boldsymbol{Q}\underline{\boldsymbol{x}} + \underline{\boldsymbol{b}}^T \underline{\boldsymbol{x}}$$

Esistono pochi casi possibili:

Se Q non è semi-definita positiva f non ha un punto di minimo.

Se Q è definita positiva $\underline{\boldsymbol{x}}^* = \boldsymbol{Q}^{-1}\underline{\boldsymbol{b}}$ è l'unico minimo globale.

Se Q è definita semi-positiva **Se Q non è singolare** $\underline{\boldsymbol{x}}^* = \boldsymbol{Q}^{-1}\underline{\boldsymbol{b}}$ è l'unico minimo globale.

Se Q è singolare non esiste una soluzione o esistono infinite soluzioni.

4 Convergenza

Ovviamente un algoritmo è buono se converge rapidamente.

4.1 Algoritmo convergente localmente e globalmente

4.1.1 Convergente localmente

Un algoritmo è **convergente localmente** quando converge solo se il punto $\underline{\boldsymbol{x}}_0$ da cui parte è in un intorno del punto ottimo $\underline{\boldsymbol{x}}^*$.

4.1.2 Convergente globalmente

Un algoritmo è **convergente globalmente** quando converge partendo da qualsiasi punto $\underline{\boldsymbol{x}}$ del dominio.

4.2 Velocità di convergenza

4.2.1 Q-lineare

Quando il rapporto tra lo step k e lo step $k+1$ mantiene un valore costante.

4.2.2 Q-super-lineare

Quando il rapporto tra lo step k e lo step $k+1$ all'infinito è nullo.

4.2.3 Q-quadratica

Quando il rapporto tra lo step k quadro e lo step $k+1$ mantiene un valore costante.

5

Metodi di ottimizzazione continua

Si tratta del problema di determinare a partire dallo step k lo step $k+1$ in modo tale da avvicinarsi all'ottimo della funzione.

Esistono due strategie:

Line search Si determina una direzione e quindi si stabilisce una distanza con cui muoversi in questa direzione.

Trust region Si determina un raggio (una distanza) di confidenza e quindi si stabilisce una direzione nel limite di questa circonferenza verso cui muoversi.

Esiste chiaramente un trade-off tra velocità e precisione nello stabilire la distanza con cui muoversi.

5.1 Condizioni di Wolfe

Per essere efficace, la line-search approssimata richiede che siano rispettate le condizioni di Wolfe. Esse sono due, una di **decremento sufficiente** ed una di **curvatura**, dove i coefficienti $\underline{\boldsymbol{c}}$ sono tali che $0 < c_1 < c_2 < 1$

$$f\left(\underline{\boldsymbol{x}} + \alpha \underline{\boldsymbol{d}}\right) \leqslant f\left(\underline{\boldsymbol{x}}\right) + c_1 \alpha \nabla f\left(\underline{\boldsymbol{x}}\right)^T \underline{\boldsymbol{d}}$$

(a) Condizione di decremento sufficiente

$$\nabla f\left(\underline{\boldsymbol{x}} + \alpha \underline{\boldsymbol{d}}\right)^T \underline{\boldsymbol{d}} \geqslant c_2 \nabla f\left(\underline{\boldsymbol{x}}\right)^T \underline{\boldsymbol{d}}$$

(b) Condizione di curvatura

Figura 5.1: Condizioni di Wolfe

Le condizioni di Wolfe forti introducono un vincolo di segno sulla curvatura (valore assoluto).

5.2 Metodo di Armijo per stabilire la stepsize

Si itera riducendo gradualmente la distanza di un fattore σ, usualmente pari circa a 0.9 sino a che il valore dello step successivo è più vicino all'ottimo dello step corrente. Nei metodi Newton e quasi Newton il coefficiente della distanza è inizializzato usualmente a 1.

5.3 Convergenza di metodi LS approssimati

Se definiamo θ_k come l'angolo tra $\underline{\boldsymbol{d}}_k$ e $-\nabla f_k$, allora possiamo determinare il coseno dell'angolo come:

$$\cos\theta_k = \frac{-\nabla f\left(\underline{\boldsymbol{x}}\right)_k^T \underline{\boldsymbol{d}}_k}{\left\|\nabla f\left(\underline{\boldsymbol{x}}\right)_k\right\| \cdot \left\|\underline{\boldsymbol{d}}_k\right\|}$$

Teorema 5.3.1 (Convergenza dei metodi di ricerca lineare approssimata). Sia $\underline{\boldsymbol{d}}_k$ una direzione di discesa e sia α_k una distanza che rispetti le condizioni di Wolfe. Sia inoltre f una funzione limitata inferiormente su $\mathbb{R}^n$, **differenziabile continuamente** sull'insieme M che contiene $L_f = \{\underline{\boldsymbol{x}} : f\left(\underline{\boldsymbol{x}}\right) \leqslant f\left(\underline{\boldsymbol{x}}_0\right)\}$, dove $\underline{\boldsymbol{x}}_0$ è il punto di inizio. Assumiamo inoltre che ∇f sia **lipschitziana** sull'insieme M. Allora:

$$\sum_{j=0}^{\infty} \cos^2\theta_j \left\|\nabla f\left(\underline{\boldsymbol{x}}_j\right)\right\|^2 \leqslant \infty$$

Dimostrazione 5.3.2. Essendo valida la **condizione di curvatura**, allora vale la disequazione:

$$\nabla f_{k+1}^T \underline{\boldsymbol{d}}_k \geqslant c_2 \nabla f_k^T \underline{\boldsymbol{d}}_k$$

Sottraggo ad ambo i lati $\nabla f_k^T \underline{\boldsymbol{d}}_k$ ed ottengo:

$$\left(\nabla f_{k+1} - \nabla f_k\right)^T \underline{\boldsymbol{d}}_k \geqslant (c_2 - 1) \nabla f_k^T \underline{\boldsymbol{d}}_k$$

Siccome il gradiente della funzione è **lipschitziano** (non va a infinito in un $\delta \underline{\boldsymbol{x}}$) vale la disequazione:

$$\left(\nabla f_{k+1}^T - \nabla f_k\right)^T \underline{\boldsymbol{d}}_k \leqslant \left\|\nabla f_{k+1}^T - \nabla f_k\right\| \left\|\underline{\boldsymbol{d}}_k\right\| \leqslant L \left\|\underline{\boldsymbol{x}}_{k+1} - \underline{\boldsymbol{x}}_k\right\| \left\|\underline{\boldsymbol{d}}_k\right\| = \alpha_k L \left\|\underline{\boldsymbol{d}}_k\right\|^2$$

Da questa disequazione ricaviamo il valore di α_k:

$$\alpha_k \geqslant \frac{c_2 - 1}{L} \frac{\nabla f_k^T \underline{\boldsymbol{d}}_k}{\left\|\underline{\boldsymbol{d}}_k\right\|^2}$$

Essendo quindi valida la **condizione di decremento sufficiente** vale la disequazione:

$$f_{k+1} \leqslant f_k + c_1 \alpha_k \underline{\boldsymbol{d}}_k^T \nabla f_k$$

Sostituendo il valore di α_k ottenuto al passaggio precedente ottengo:

$$f_{k+1} \leqslant f_k - c_1 \frac{1 - c_2}{L} \frac{\left(\nabla f_k^T \underline{\boldsymbol{d}}_k\right)^2}{\left\|\underline{\boldsymbol{d}}_k\right\|^2}$$

Pongo $c = c_1 \frac{1-c_2}{L}$ ed ottengo:

$$f_{k+1} \leqslant f_k - c \frac{\left(\nabla f_k^T \underline{\boldsymbol{d}}_k\right)^2}{\left\|\underline{\boldsymbol{d}}_k\right\|^2}$$

Sostituisco $\cos^2 \theta_k = \frac{\left(-\nabla f(\underline{\boldsymbol{x}})_k^T \underline{\boldsymbol{d}}_k\right)^2}{\left\|\nabla f(\underline{\boldsymbol{x}})_k\right\|^2 \cdot \left\|\underline{\boldsymbol{d}}_k\right\|^2}$ ed ottengo:

$$f_{k+1} \leqslant f_k - c \cos^2 \theta_k \left\|\nabla f_k\right\|^2$$

Per ricorsione ottengo la sommatoria:

$$f_{k+1} \leqslant f_0 - c \sum_{j=0}^{k} \cos^2 \theta_j \left\|\nabla f_j\right\|^2$$

Siccome la funzione f è limitata inferiormente si ottiene la **condizione di Zoutendijk**:

$$c \sum_{j=0}^{k} \cos^2 \theta_j \left\|\nabla f_j\right\|^2 \leqslant f_0 - f_{k+1} < \infty$$

Questo implica che:

$$\lim_{k \to \infty} \cos^2 \theta_k \left\|\nabla f_k\right\|^2 \to 0$$

Quindi se l'algoritmo soddisfa anche la **condizione angolare** (cioè sceglie una direzione di discesa che la rispetta) $\cos \theta_k \geqslant \epsilon > 0$ allora **converge**:

$$\lim_{k \to \infty} \left\|\nabla f\left(\underline{\boldsymbol{x}}_k\right)\right\| = 0$$

□

6

Metodi di ottimizzazione

6.1 Metodi a discesa rapida

Son metodi che usano come funzione di aggiornamento $\underline{\boldsymbol{x}}_{k+1} = \underline{\boldsymbol{x}}_k - \alpha_k \nabla f_k$, in cui le direzioni sono ortogonali al contorno della funzione. Non dovendo calcolare l'hessiana lo sforzo computazionale non è eccessivo e converge globalmente, ma estremamente piano quando una funzione è patologica.

Teorema 6.1.1 (Velocità di convergenza locale dei metodi a discesa rapida). Data una matrice $\boldsymbol{Q}$ definita positiva, la seguente relazione vale $\forall \underline{\boldsymbol{x}} \in \mathbb{R}^n$:

$$\frac{\left(\underline{\boldsymbol{x}}^T \underline{\boldsymbol{x}}\right)^2}{\left(\underline{\boldsymbol{x}}^T \boldsymbol{Q} \underline{\boldsymbol{x}}\right)\left(\underline{\boldsymbol{x}}^T \boldsymbol{Q}^{-1} \underline{\boldsymbol{x}}\right)} \geqslant \frac{\lambda_{\min}\lambda_{\max}}{(\lambda_{\min}+\lambda_{\max})^2}$$

Dove $\lambda_{\min}$ e $\lambda_{\max}$ son gli autovalori minimo e massimo di $\boldsymbol{Q}$.

Ne segue che la velocità di convergenza dei metodi a discesa rapida è **lineare** per i modelli quadratici.

$$\frac{\left\|\underline{\boldsymbol{x}}_{k+1} - \underline{\boldsymbol{x}}^*\right\|_{\boldsymbol{Q}}}{\left\|\underline{\boldsymbol{x}}_k - \underline{\boldsymbol{x}}^*\right\|_{\boldsymbol{Q}}} \leqslant \frac{(\lambda_{\max} - \lambda_{\min})}{(\lambda_{\max} + \lambda_{\min})}$$

6.2 Metodi Newton

Prendendo in considerazione l'approssimazione di Taylor fermata al secondo ordine, quindi con l'hessiana, otteniamo come direzione di discesa quella che minimizza $\nabla f^T \underline{\boldsymbol{d}} + \frac{1}{2}\underline{\boldsymbol{d}}^T \boldsymbol{H}\underline{\boldsymbol{d}}$, cioè $\underline{\boldsymbol{d}} = -\boldsymbol{H}^{-1}\nabla f$. Quando $\boldsymbol{H}$ è **definita positiva** vale che:

$$\nabla f^T \underline{\boldsymbol{d}} = -\underline{\boldsymbol{d}}^T \boldsymbol{H}\underline{\boldsymbol{d}} \leqslant -\sigma \left\|\underline{\boldsymbol{d}}\right\|^2$$

Cioè quando $\boldsymbol{H}$ è **definita positiva** la direzione di Newton è la **direzione di discesa**.

Nei **modelli quadratici** con $\boldsymbol{Q}$ **definita positiva** il metodo di Newton converge in un'iterazione, altrimenti non converge. Su funzioni generiche, la qualità della direzione dipende da quanto è definita positiva la matrice hessiana.

Teorema 6.2.1 (Convergenza dei metodi di Newton). Sia $f \in C^2$ e sia $\boldsymbol{H}\left(\underline{\boldsymbol{x}}\right)$ continuamente lipschitziana in un intorno del punto ottimo $\underline{\boldsymbol{x}}^*$. Si assuma che valga $\underline{\boldsymbol{x}}_{k+1} = \underline{\boldsymbol{x}}_k + \underline{\boldsymbol{d}}_k$. Allora:

1. Se $\underline{\boldsymbol{x}}_0$ è sufficientemente vicino a $\underline{\boldsymbol{x}}^*$, allora $\left\{\underline{\boldsymbol{x}}_k\right\} \to \underline{\boldsymbol{x}}^*$
2. $\left\{\underline{\boldsymbol{x}}_k\right\}$ converge **quadraticamente**
3. $\left\{\left\|\nabla f\left(\underline{\boldsymbol{x}}_k\right)\right\|\right\}$ converge quadraticamente a zero.

I metodi Newton sono **convergenti localmente**.

La complessità computazionale è di $O(n^3)$

6.2.1 Metodi Newton modificati

Sono metodi che modificano l'Hessiano, o rendendo la matrice positiva o scegliendo una direzione di discesa tramite metodi di discesa rapida quando necessario.

6.3 Metodi Quasi-Newton

Sono metodi in cui viene utilizzata un'approssimazione dell'Hessiano, che è computazionalmente costoso da calcolare. Viene utilizzata al suo posto una matrice chiamata $\boldsymbol{G}_k$ al posto di $\boldsymbol{H}_k^{-1}$, e quindi calcolano la direzione di discesa come $\underline{\boldsymbol{d}}_k = -\boldsymbol{G}_k \nabla f\left(\underline{\boldsymbol{x}}_k\right)$.

Definizione 6.3.1 (Relazione secante). Definendo $\underline{\boldsymbol{p}}_k = \nabla f(\underline{\boldsymbol{x}}_{k+1}) - \nabla f\left(\underline{\boldsymbol{x}}_k\right)$ possiamo definire la relazione secante:

$$\boldsymbol{H}\left(\underline{\boldsymbol{x}}_k\right)\underline{\boldsymbol{h}}_k \approx \underline{\boldsymbol{p}}_k \qquad \vee \qquad \boldsymbol{H}\left(\underline{\boldsymbol{x}}_k\right)^{-1}\underline{\boldsymbol{p}}_k \approx \underline{\boldsymbol{h}}_k$$

Inizializzando $\boldsymbol{G}_0 = I$, possiamo imporre che ad ogni iterazione la matrice $\boldsymbol{G}_{k+1}$ debba soddisfare la relazione secante con la seguente uguaglianza:

$$\boldsymbol{G}_{k+1}\underline{\boldsymbol{p}}_k = \underline{\boldsymbol{h}}_k$$

La realizzazione specifica di come si ottiene $\boldsymbol{G}_{k+1}$ partendo da $\boldsymbol{G}_k$ varia dai differenti metodi Quasi-Newton. Questi metodi impongono che $\boldsymbol{G}_k = \boldsymbol{G}_k^T$ e che $\boldsymbol{G}_{k+1} - \boldsymbol{G}_k$ abbia un rango basso.

Ne esistono di due categorie:

1. Matrice a rango unitario simmetrico o SR1.
2. A rango due (i nomi sono le iniziali degli autori):
   (a) DFP
   (b) BFGS

I **metodi a rango due** hanno cinque proprietà interessanti:

1. La matrice $\boldsymbol{G}_k$ converge a $\boldsymbol{H}\left(\underline{\boldsymbol{x}}_k\right)^{-1}$ sui modelli quadratici, cioè va ad approssimare l'inversa della matrice hessiana.
2. Se $\boldsymbol{G}_0$ è definita positiva allora tutte le $\boldsymbol{G}_k$ sono definite positive.
3. Il costo computazionale è di $O\left(n^2\right)$ in ogni iterazione.
4. La velocità di convergenza è **superlineare**.
5. In particolare il metodo BFGS garantisce convergenza globale se lo step-size rispetta le condizioni di Wolfe.

Con **Famiglia di Broyden** si intende una famiglia di formule ottenute tramite la combinazione convessa dell'aggiornamento DFP e BFGS.

6.4 Metodi del gradiente coniugato

Si tratta di una delle tecniche più utili per risolvere sistemi lineari di grandi dimensioni e può essere adattata per risolvere problemi di ottimizzazione non lineare. Si tratta di un metodo iterativo che risolve sistemi della forma $\boldsymbol{A}\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}$ dove $\boldsymbol{A}$ è una matrice definita positiva che è $\boldsymbol{A} = \boldsymbol{A}^T$.

Definizione 6.4.1 (Vettori coniugati). Un insieme di vettori $\underline{\boldsymbol{p}}_0, \underline{\boldsymbol{p}}_1, \ldots, \underline{\boldsymbol{p}}_h$ è detto **coniugato** rispetto ad una matrice $\boldsymbol{A}$ se vale la relazione:

$$\underline{\boldsymbol{p}}_i^T \boldsymbol{A} \underline{\boldsymbol{p}}_j = 0 \quad \forall i \neq j$$

Dato un punto di inizio $\underline{\boldsymbol{x}}_0$ e le **direzioni coniugate** $\underline{\boldsymbol{p}}_0, \ldots, \underline{\boldsymbol{p}}_{n-1}$ generiamo la sequenza $\underline{\boldsymbol{x}}_{k+1} = \underline{\boldsymbol{x}}_k + \alpha_k \underline{\boldsymbol{p}}_k$ dove α_k è un minimizzatore monodimensionale ed è dato da:

$$\alpha_k = -\frac{r_k^T \underline{\boldsymbol{p}}_k}{\underline{\boldsymbol{p}}_k^T \boldsymbol{A} \underline{\boldsymbol{p}}_k}$$

Teorema 6.4.2 (Convergenza di sequenza con vettori coniugati). Dato un qualsiasi $\underline{\boldsymbol{x}}_0$, la sequenza generata da $\underline{\boldsymbol{x}}_{k+1} = \underline{\boldsymbol{x}}_k + \alpha_k \underline{\boldsymbol{p}}_k$ converge a $\underline{\boldsymbol{x}}^*$ in al più n iterazioni.

Il metodo del gradiente coniugato è un metodo a direzione coniugata con una proprietà importante: $\underline{\boldsymbol{p}}_k$ può essere ottenuta sapendo solo $\underline{\boldsymbol{p}}_{k-1}$ e $\underline{\boldsymbol{p}}_k$ risulta coniugato a tutte le direzioni precedenti.

Il vettore $\underline{\boldsymbol{p}}_k$ è definito come una combinazione lineare del gradiente e del vettore precedente:

$$\underline{\boldsymbol{p}}_k = -\nabla f_k + \beta_k \underline{\boldsymbol{p}}_{k-1}$$

Dove β è definito sfruttando la definizione di vettore coniugato:

$$\beta_k = \frac{\nabla f_k^T \boldsymbol{A} \underline{\boldsymbol{p}}_{k-1}}{\underline{\boldsymbol{p}}_{k+1}^T \boldsymbol{A} \underline{\boldsymbol{p}}_{k-1}}$$

Inizialmente il valore del vettore è $\underline{\boldsymbol{p}}_0 = -\nabla f_0$.

Esistono vari metodi del gradiente coniugato che variano in base a come viene calcolato il valore di β_{k+1} a partire dall'iterazione precedente.

In particolare è utile perché utilizza poco spazio, non è computazionalmente pesante e converge rapidamente. Viene però preferita l'eliminazione gaussiana quando il problema è di piccola dimensione, essendo meno sensibile agli errori di arrotondamento.

6.5 Metodi Trust Region

Questi metodi realizzano un modello della funzione con lo sviluppo di Taylor fermato al **secondo ordine**. Si può usare sia la matrice hessiana che una sua approssimazione.

Tramite questo si determina un raggio della trust region e quindi si procede ad ottenere una direzione tale che minimizzi la funzione, e nel caso la discesa non sia sufficiente si riduce il raggio e si ripete l'operazione. Per un raggio sufficientemente grande, quando viene utilizzata l'hessiana e questa è definita positiva la direzione che si ottiene coincide con quella che si otterrebbe dai metodi Newton.

Il metodo della trust region richiede di calcolare una sequenza di sotto-problemi la cui funzione obbiettivo e vincoli sono entrambi quadratici.

Teorema 6.5.1. Il vettore $\underline{\boldsymbol{p}}^*$ è una soluzione globale per il problema di trust-region di raggio Δ:

$$\min_{\underline{\boldsymbol{p}} \in \mathbb{R}^n} m(\underline{\boldsymbol{p}}) = f\left(\underline{\boldsymbol{x}}\right) + \nabla f^T \underline{\boldsymbol{p}} + \frac{1}{2}\underline{\boldsymbol{p}}^T \boldsymbol{B}\underline{\boldsymbol{p}} \qquad \left\|\underline{\boldsymbol{p}}\right\| \leqslant \Delta$$

se e solo se $\underline{\boldsymbol{p}}^*$ appartiene alla trust-region ed esiste uno scalare $\lambda \geqslant 0$ tale per cui:

$$\begin{cases} (\boldsymbol{B} + \lambda \boldsymbol{I})\underline{\boldsymbol{p}}^* & = -\nabla f \\ \lambda(\Delta - \left\|\underline{\boldsymbol{p}}^*\right\|) & = 0 \\ (\boldsymbol{B} + \lambda \boldsymbol{I}) & \text{è definita semipositiva} \end{cases}$$

I metodi trust-region garantiscono convergenza globale grazie ad una riduzione sufficiente del modello che può essere quantificata in termini del **punto di Cauchy**, definito come il punto minimo del modello quadratico lungo la direzione $-\nabla f$. Questo punto può essere calcolato nell'ordine di $O\left(n^2\right)$ e viene utilizzato per determinare se la soluzione di un sotto problema è accettabile.

Un metodo trust-region sarà globalmente convergente se un suo step $\underline{\boldsymbol{p}}_k$ da un riduzione nel modello m_k che è almeno un multiplo positivo della decrescita ottenuta tramite lo step di Cauchy. Se scegliessimo sempre il punto di Cauchy per la discesa implementeremmo banalmente il metodo di discesa rapida con una scelta fissata della step length e questo sarebbe poco efficiente.

Dobbiamo utilizzare le informazioni che possiamo trarre dalla matrice $\boldsymbol{B}_k$, la nostra approssimazione (o coincidente) della matrice hessiana.

6.5.1 Metodo zampa di cane

Questo metodo risulta applicabile quando la matrice $\boldsymbol{B}$ è definita positiva. Si inizia computando il punto $\underline{\boldsymbol{p}}^{(B)} = -\boldsymbol{B}^{-1}\nabla f$ e quindi se questo punto cade nel raggio della regione ammissibile esso coincide con il punto ottimo $\underline{\boldsymbol{p}}^*$. Altrimenti si determina una traiettoria ottimale tra un punto $\underline{\boldsymbol{p}}^{(U)} = -\frac{\nabla f^T \nabla f}{\nabla f^T \boldsymbol{B} \nabla f}\nabla f$ ed il punto $\underline{\boldsymbol{p}}^{(B)}$ seguendo la curva seguente:

$$\underline{\boldsymbol{p}}(\tau) = \begin{cases} \tau \underline{\boldsymbol{p}}^{(U)} & \tau \in [0,1] \\ \underline{\boldsymbol{p}}^{(U)} + (\tau - 1)(\underline{\boldsymbol{p}}^{(B)} - \underline{\boldsymbol{p}}^{(U)}) & \tau \in [1,2] \end{cases}$$

6.5.2 Proprietà della matrice B

Se la matrice $\boldsymbol{B}$ risulta definita semi positiva, allora:

1. $\left\|\underline{\boldsymbol{p}}(\tau)\right\|$ è una funzione crescente di τ
2. $m(\underline{\boldsymbol{p}}(\tau))$ è una funzione decrescente di τ
3. τ può essere calcolata risolvendo l'equazione scalare quadratica:

$$\left\|\underline{\boldsymbol{p}}^{(U)} + (\tau - 1)(\underline{\boldsymbol{p}}^{(B)} - \underline{\boldsymbol{p}}^{(U)})\right\|^2 = \Delta^2$$

4. Esistono metodi iterativi per risolvere il modello quadratico.
5. La convergenza può arrivare ad essere **super-lineare**.

7

Ottimizzazione vincolata

Un problema è detto di ottimizzazione vincolata quando il dominio e limitato da una o più funzioni. In presenza di vincoli, problemi complessi possono divenire semplici: funzioni non convesse su $\mathbb{R}$ per esempio potrebbero esserlo sul dominio di definizione.

Le condizioni di ottimalità possono essere ottenute tramite la funzione lagrangiana.

Teorema 7.0.1. Data una funzione $f(\underline{\boldsymbol{x}})$ e $\underline{\boldsymbol{h}}$ vincoli bilateri, con $f, \underline{\boldsymbol{h}} \in C^1$. Se i gradienti dei vincoli nel punto ottimo $\underline{\boldsymbol{x}}^*$ sono **linearemente indipendenti**, se $\underline{\boldsymbol{x}}^*$ sono di minimo locale di $f(\underline{\boldsymbol{x}})$ ed esso soddisfa i vincoli $\underline{\boldsymbol{h}}$, allora esiste un vettore $\underline{\boldsymbol{\lambda}}^*$ tale che $(\underline{\boldsymbol{x}}^*, \underline{\boldsymbol{\lambda}}^*)$ è un punto stazionario della funzione Lagrangiana L:

$$\begin{cases} \frac{\partial L}{\partial x_i} = \frac{\partial f(\underline{\boldsymbol{x}}^*)}{\partial x_i} + \sum_{j=1}^{h} \lambda_j^* \frac{\partial h_j(\underline{\boldsymbol{x}}^*)}{\partial x_i} = 0 \\ \frac{\partial L}{\partial \lambda_i} = h_j(\underline{\boldsymbol{x}}^*) = 0 \end{cases}$$

Definizione 7.0.2 (Condizioni di regolarità). Un punto $\underline{\boldsymbol{x}}^*$ soddisfa le condizioni di regolarità se non esiste un vettore $\underline{\boldsymbol{h}}$ tale per cui tutti i gradienti dei vincoli attivi sono linearmente indipendenti in quel punto. Un punto che soddisfa queste condizioni è detto **regolare**.

Teorema 7.0.3 (Condizioni sufficienti del prim'ordine per problemi convessi). Data una funzione $f(\underline{\boldsymbol{x}})$ ed un vettore $\underline{\boldsymbol{h}}$ di vincoli bilateri, con $f, \underline{\boldsymbol{h}} \in C^1$. Se la matrice jacobiana $J(\underline{\boldsymbol{x}}^*)$ ha rango massimo $\|\underline{\boldsymbol{h}}\|$, se esiste un vettore $\underline{\boldsymbol{\lambda}}^*$ tale che $\underline{\boldsymbol{x}}^*, \underline{\boldsymbol{\lambda}}^*$ è un punto stazionario della funzione lagrangiana L, allora $\underline{\boldsymbol{x}}^*$ è un minimo locale di $f(\underline{\boldsymbol{x}})$.

Mettendo assieme tutti questi risultati matematici otteniamo le **condizioni di Karush-Kuhn-Tucker**, o condizioni KKT

Teorema 7.0.4 (Condizioni di Karush-Kuhn-Tucker). Sia f una funzione, h_i con $i \in \{1, \dots, s\}$ dei vincoli bilateri e g_j con $j \in \{1, \dots, m\}$ dei vincoli monolateri e sia l'insieme X definito come:

$$X = \{x \in \mathbb{R}^n : g_j(x) \leqslant 0, h_i(x) = 0 \quad \forall i, j\} \quad \text{e} \quad f, g_j, h_i \in C^1(X) \quad \forall i, j$$

Se x^* è un punto regolare in X e un punto di minimo locale per $f \in X$, allora esistono s moltiplicatori $\lambda_i \in \mathbb{R}$ e m $\mu_j \geqslant 0$ tali che:

$$\nabla f(x^*) + \sum_{i=1}^{s} \lambda_i \nabla h_i(x^*) + \sum_{j=1}^{m} \mu_j \nabla g_j(x^*) = 0$$

$$\mu_j g_j(x^*) = 0$$

Teorema 7.0.5 (Condizioni KKT nel caso convesso). Se le funzioni $f, \underline{\boldsymbol{g}}, \underline{\boldsymbol{h}} \in C^1$ sono **funzioni convesse** le condizioni KKT sono condizioni sufficienti.

Definizione 7.0.6 (Cono critico). Dato un punto $\underline{\boldsymbol{x}}^*$ ed i vettori di moltiplicatori $\underline{\boldsymbol{\lambda}}^*$ e $\underline{\boldsymbol{\mu}}^*$ che soddisfano le condizioni KKT, viene chiamato **cono critico** l'insieme delle direzioni di discesa verso il punto $\underline{\boldsymbol{x}}^*$.

$$C(\underline{\boldsymbol{x}}^*, \underline{\boldsymbol{\lambda}}^*, \underline{\boldsymbol{\mu}}^*) = \left\{ \underline{\boldsymbol{d}} \in F(\underline{\boldsymbol{x}}^*) : \nabla h_j(\underline{\boldsymbol{x}}^*)^T \underline{\boldsymbol{d}} = 0, j \in E; g_j(\underline{\boldsymbol{x}}^*)^T \underline{\boldsymbol{d}}, j \in I, \text{ con } \underline{\boldsymbol{\lambda}}^* > 0 \right\}$$

Queste direzioni sono ortogonali al gradiente di f.

7.1 Condizioni di ottimilità del secondo ordine

Teorema 7.1.1 (Matrice hessiana semi-definita positiva nel punto minimo). Dato un problema di ottimizzazione in cui le funzioni $f, \underline{\boldsymbol{g}}, \underline{\boldsymbol{h}} \in C^2$. Se $\underline{\boldsymbol{x}}^*$ è un minimo locale che **appartiene alla regione ammissibile** ed esistono due vettori $\underline{\boldsymbol{\lambda}}^*$ e $\underline{\boldsymbol{\mu}}^*$ che soddisfano le condizioni KKT, allora:

$$\underline{\boldsymbol{d}}^T \nabla^2_{\underline{\boldsymbol{x}},\underline{\boldsymbol{x}}} L(\underline{\boldsymbol{x}}^*, \underline{\boldsymbol{\lambda}}^*, \underline{\boldsymbol{\mu}}^*) \underline{\boldsymbol{d}} \geqslant 0, \qquad \forall \underline{\boldsymbol{d}} \in \text{Cono critico}$$

Se la matrice hessiana è **definita positiva** nel cono critico, allora le condizioni KKT divengono sufficienti.

Teorema 7.1.2 (Punto di minimo locale stretto). Dato un problema di ottimizzazione in cui le funzioni $f, \underline{\boldsymbol{g}}, \underline{\boldsymbol{h}} \in C^2$. Se $\underline{\boldsymbol{x}}^*$ appartiene alla regione ammissibile, esistono due vettori $\underline{\boldsymbol{\lambda}}^*$ e $\underline{\boldsymbol{\mu}}^*$ che soddisfano le condizioni KKT e vale la seguente relazione:

$$\underline{\boldsymbol{d}}^T \nabla^2_{\underline{\boldsymbol{x}},\underline{\boldsymbol{x}}} L(\underline{\boldsymbol{x}}^*, \underline{\boldsymbol{\lambda}}^*, \underline{\boldsymbol{\mu}}^*) \underline{\boldsymbol{d}} > 0, \qquad \forall \underline{\boldsymbol{d}} \in \text{Cono critico}$$

Allora $\underline{\boldsymbol{x}}^*$ è un punto di minimo locale stretto.

7.2 Modello quadratico con vincoli lineari

Un problema quadratico del tipo:

$$\begin{aligned} \min q\left(\underline{\boldsymbol{x}}\right) &= \frac{1}{2}\underline{\boldsymbol{x}}^T Q \underline{\boldsymbol{x}} + \underline{\boldsymbol{b}}^T \underline{\boldsymbol{x}} \\ \boldsymbol{A}\underline{\boldsymbol{x}} &\geqslant \underline{\boldsymbol{d}} \end{aligned}$$

è risolvibile facilmente quando la matrice $\boldsymbol{Q}$ è definita positiva.

7.2.1 Metodo dell'insieme attivo primale per problemi quadratici convessi (Primal Active set)

I metodi **Primal Active set** ad ogni iterazione risolvono un sottoproblema quadratico in cui i vincoli di disuguaglianza sono imposti come uguaglianze sul **working set** W_k.

I gradienti $\underline{\boldsymbol{\alpha}}^T{}_i$ dei vincoli sul set W_k sono linearmente indipendenti.

Per prima cosa viene controllato se $\underline{\boldsymbol{x}}_k$ minimizza la funzione quadratica $q\left(\underline{\boldsymbol{x}}\right)$ su W_k e se questo non avviene viene calcolato uno step p risolvendo un sotto-problema quadratico sul set W_k, vincolato da uguaglianze:

$$\begin{aligned} \underline{\boldsymbol{p}} &= \underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}_k \\ \underline{\boldsymbol{g}}_k &= Q\underline{\boldsymbol{x}}_k + \underline{\boldsymbol{b}} \\ \underline{\boldsymbol{p}}_k &= \operatorname{argmin} q\left(\underline{\boldsymbol{x}}\right) = \operatorname{argmin} q(\underline{\boldsymbol{x}}_k + p) = \operatorname{argmin} \frac{1}{2}\underline{\boldsymbol{p}}^T Q \underline{\boldsymbol{p}} + \underline{\boldsymbol{g}}^T{}_k \underline{\boldsymbol{p}} + \rho_k \\ \underline{\boldsymbol{\rho}}_k &= \frac{1}{2}\underline{\boldsymbol{x}}^T{}_k Q \underline{\boldsymbol{x}}_k + \underline{\boldsymbol{b}}^T \underline{\boldsymbol{x}}_k \\ \underline{\boldsymbol{\alpha}}^T{}_i \underline{\boldsymbol{p}} &= 0 \qquad \forall i \in W_q \end{aligned}$$

Per ogni $i \in W_k$ vale che:

$$\underline{\boldsymbol{\alpha}}^T{}_i(\underline{\boldsymbol{x}}_k + \alpha \underline{\boldsymbol{p}}_k) = \underline{\boldsymbol{\alpha}}^T{}_i \underline{\boldsymbol{x}}_k = \underline{\boldsymbol{d}}_i \forall \alpha$$

Se $\underline{\boldsymbol{p}}_k \neq 0$ allora aggiorno il valore di α_k ed ottengo il prossimo punto, $\underline{\boldsymbol{x}}_{k+1} = \underline{\boldsymbol{x}}_k + \alpha_k \underline{\boldsymbol{p}}_k$:

$$\alpha_k = \min\left\{1, \min\left\{\frac{\underline{\boldsymbol{d}}_i - \underline{\boldsymbol{\alpha}}^T{}_i \underline{\boldsymbol{x}}_k}{\underline{\boldsymbol{\alpha}}^T{}_i \underline{\boldsymbol{p}}_k} : i \neq W_k, \underline{\boldsymbol{\alpha}}^T{}_i \underline{\boldsymbol{p}}_k < 0\right\}\right\}$$

Altrimenti testo le condizioni KKT o aggiorno il set di lavoro W_k.

8 Algoritmi

8.1 Metodo della penalità quadratica

Questo metodo trasforma il problema vincolato in uno privo di vincoli, sfruttando una **penalty function**.

Si parte quindi da un problema vincolato:

$$\begin{aligned} &\min f\left(\underline{\boldsymbol{x}}\right) \\ &\underline{\boldsymbol{h}}\left(\underline{\boldsymbol{x}}\right) = \underline{\mathbf{0}} \end{aligned}$$

Si costruisce una **penalty function** utilizzando i vincoli:

$$p\left(\underline{\boldsymbol{x}}\right) = \sum_{j=1}^{h} h_j^2\left(\underline{\boldsymbol{x}}\right)$$

Figura 8.1: Penalty function

Il modello che si ottiene è quindi la somma pesata tra la funzione obbiettivo e la **penalty function**:

$$\min q\left(\underline{\boldsymbol{x}}\right) = f\left(\underline{\boldsymbol{x}}\right) + \alpha \sum_{j=1}^{h} h_j^2\left(\underline{\boldsymbol{x}}\right)$$

Aumentando il parametro α a $+\infty$, aumentiamo la penalità della violazione dei vincoli, aumentandone la severità.

Possiamo usare su questo problema le tecniche tratte dall'ottimizzazione svincolata.

8.2 Metodo delle barriere

Dato un problema vincolato con disuguaglianze:

$$\min f\left(\underline{\boldsymbol{x}}\right)$$
$$\underline{\boldsymbol{g}}\left(\underline{\boldsymbol{x}}\right) \leqslant \underline{\mathbf{0}}$$

Si procede dividendo la regione ammissibile in due set: uno definito dalla frontiera dei vincoli ($\underline{\boldsymbol{g}}\left(\underline{\boldsymbol{x}}\right) = \underline{\mathbf{0}}$) ed uno dall'interno ($\underline{\boldsymbol{g}}\left(\underline{\boldsymbol{x}}\right) < \underline{\mathbf{0}}$).

Questo metodo è applicabile solo quando il set interno non è vuoto. Viene utilizzata una **funzione di barriera** che aumenta a ∞ quando il punto $\underline{\boldsymbol{x}}$ tende al set di frontiera.

$$v\left(\underline{\boldsymbol{x}}\right) = -\sum_{j=1}^{k} \log\left(-g_j\left(\underline{\boldsymbol{x}}\right)\right)$$

Figura 8.2: Barrier function

Mettendole assieme si ottiene:

$$\min q\left(\underline{\boldsymbol{x}}\right) = f\left(\underline{\boldsymbol{x}}\right) - \alpha\, v\left(\underline{\boldsymbol{x}}\right)$$

8.3 Metodo della proiezione del gradiente

Viene applicato su problemi con vincoli lineari:

$$\min f\left(\underline{\boldsymbol{x}}\right)$$
$$\boldsymbol{A}\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}$$

Si inizia con una soluzione ammissibile $\underline{\boldsymbol{x}}' : \boldsymbol{A}\underline{\boldsymbol{x}}' = \underline{\boldsymbol{b}}$ e si cerca una soluzione migliore $\underline{\boldsymbol{x}} = \underline{\boldsymbol{x}}' + \alpha\underline{\boldsymbol{d}}$ lungo la direzione $\underline{\boldsymbol{d}}$ che deve essere normalizzata, deve minimizzare la derivata direzionale $\nabla f(\underline{\boldsymbol{x}}')^T\underline{\boldsymbol{d}}$ in $\underline{\boldsymbol{x}}'$ ed il punto $\underline{\boldsymbol{x}}$ deve essere ammissibile, cioè $\boldsymbol{A}\underline{\boldsymbol{d}} = \underline{\boldsymbol{0}}$.

Da questo si va a costruire il problema di minimizzazione della derivata direzionale, da cui si ottiene che la direzione $\underline{\boldsymbol{d}}$ migliore è:

$$\underline{\boldsymbol{d}} = \frac{\left(\boldsymbol{I} - \boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{A}^T\right)^{-1}\boldsymbol{A}\right)\nabla f(\underline{\boldsymbol{x}}')}{\left\|\left(\boldsymbol{I} - \boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{A}^T\right)^{-1}\boldsymbol{A}\right)\nabla f(\underline{\boldsymbol{x}}')\right\|}$$

La direzione che migliora di più $f\left(\underline{\boldsymbol{x}}\right)$ è l'anti-gradiente e $\underline{\boldsymbol{d}}$ risulta esserne la **proiezione** sull'iperpiano $\boldsymbol{A}\underline{\boldsymbol{x}} = \underline{\boldsymbol{d}}$. La matrice $P = \left(I - \boldsymbol{A}^T\left(\boldsymbol{A}\boldsymbol{A}^T\right)^{-1}\boldsymbol{A}\right)\nabla f(\underline{\boldsymbol{x}}')$ è detta **matrice di proiezione**.

Si procede quindi con $\underline{\boldsymbol{d}} = -P\nabla f\left(\underline{\boldsymbol{x}}_k\right)$ e si ottiene il punto successivo tramite:

$$\underline{\boldsymbol{x}}_{k+1} = \underline{\boldsymbol{x}}_k + \alpha\underline{\boldsymbol{d}}$$

Dove α può essere ottenuto con per esempio il metodo di Armijo.

8.3.1 Caso con vincoli generici non lineari

Nel caso con vincoli generici viene utilizzato uno sviluppo di Taylor per individuare dei vincoli lineari nell'intorno del punto corrente $\underline{\boldsymbol{x}}'$.

Partendo quindi da un problema del tipo:

$$\min f\left(\underline{\boldsymbol{x}}\right)$$
$$\underline{\boldsymbol{h}}\left(\underline{\boldsymbol{x}}\right) = \underline{\boldsymbol{0}}$$

Si svolge lo sviluppo di Taylor:

$$h_j\left(\underline{\boldsymbol{x}}\right) = h_j(\underline{\boldsymbol{x}}') + \nabla h_j(\underline{\boldsymbol{x}}')^T(\underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}')$$

Da cui si ottiene:

$$\nabla h_j(\underline{\boldsymbol{x}}')^T\underline{\boldsymbol{x}} - \nabla h_j(\underline{\boldsymbol{x}}')^T\underline{\boldsymbol{x}}' = 0$$

Imponendo quindi $\boldsymbol{A} = \nabla h_j(\underline{\boldsymbol{x}}')^T$ e $\underline{\boldsymbol{b}} = \nabla h_j(\underline{\boldsymbol{x}}')^T\underline{\boldsymbol{x}}'$ ci si riconduce al caso precedente:

$$\min f\left(\underline{\boldsymbol{x}}\right)$$
$$\boldsymbol{A}\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}$$

Con l'eccezione che ora la matrice di proiezione $\boldsymbol{P}$ dipende da $\boldsymbol{A}$, che a sua volta ora dipende da $\underline{\boldsymbol{x}}'$. Pertanto la direzione da usare sarà $\underline{\boldsymbol{d}} = -\boldsymbol{P}\left(\underline{\boldsymbol{x}}_k\right)\nabla f\left(\underline{\boldsymbol{x}}_k\right)$.

Per costruzione del metodo però, molto probabilmente il punto ottenuto $\underline{\boldsymbol{x}}_{k+1}$ non soddisfa i vincoli non lineari originali, per cui si procede con uno step correttivo ottenendo $\underline{\boldsymbol{x}}_{k+1_{\text{corretto}}}$ con:

$$P\left(\underline{\boldsymbol{x}}_k\right)\left(\underline{\boldsymbol{x}}_{k+1_{\text{corretto}}} - \underline{\boldsymbol{x}}_{k+1}\right) = \underline{\boldsymbol{0}} \quad \wedge \quad \underline{\boldsymbol{h}}(\underline{\boldsymbol{x}}_{k+1_{\text{corretto}}}) = 0$$

Da cui si ottiene che $\underline{\boldsymbol{x}}_{k+1_{\text{corretto}}}$ risulta:

$$\underline{\boldsymbol{x}}_{k+1_{\text{corretto}}} \approx \underline{\boldsymbol{x}}_{k+1} - \boldsymbol{A}^T\left(A\boldsymbol{A}^T\right)^{-1}\underline{\boldsymbol{h}}(\underline{\boldsymbol{x}}_{k+1})$$

Lo step correttivo viene applicato fino a che il valore di $\underline{\boldsymbol{h}}(\underline{\boldsymbol{x}}_{k+1})$ è sufficientemente piccolo, mentre l'algoritmo complessivo si interrompe quando $P\left(\underline{\boldsymbol{x}}_k\right)\nabla f\left(\underline{\boldsymbol{x}}_k\right) \approx 0$.

8.4 Augmented lagrangean method

Questo metodo combina l'uso di una funzione lagrangiana con una funzione di penalità quadratica, l'idea è quella di approssimare i moltiplicatori di Lagrange. Si inizia con un problema generico:

$$\min f\left(\underline{\boldsymbol{x}}\right)$$
$$\underline{\boldsymbol{h}}\left(\underline{\boldsymbol{x}}\right) = \underline{\mathbf{0}}$$

E si realizza una **funzione lagrangiana incrementata**:

$$L(\underline{\boldsymbol{x}}, \underline{\boldsymbol{\lambda}}, \rho) = f\left(\underline{\boldsymbol{x}}\right) + \sum_{j=1}^{h} \lambda_j h_j\left(\underline{\boldsymbol{x}}\right) + \rho \sum_{j=1}^{h} h_j^2\left(\underline{\boldsymbol{x}}\right)$$

Figura 8.3: Augmented lagrangean function

1. Nel caso in cui $\lambda_j = 0$ si ottiene la funzione di penalità.
2. Se è noto $\lambda^* \forall \rho > 0$ minimizzando $L(\underline{\boldsymbol{x}}, \underline{\boldsymbol{\lambda}}, \rho)$ rispetto a $\underline{\boldsymbol{x}}$ si ottiene $\underline{\boldsymbol{x}}^*$
3. Se $\underline{\boldsymbol{\lambda}}^k$ è una approssimazione valida di $\underline{\boldsymbol{\lambda}}^*$, allora possiamo approssimare $\underline{\boldsymbol{x}}^*$ minimizzando $L(\underline{\boldsymbol{x}}, \underline{\boldsymbol{\lambda}}^k, \rho)$ anche per valori bassi di ρ.
4. Il parametro ρ deve garantire che $L(\underline{\boldsymbol{x}}, \underline{\boldsymbol{\lambda}}^k, \rho)$ abbia un minimo locale rispetto a $\underline{\boldsymbol{x}}$ e non solamente un punto stazionario.

Si procede quindi iterativamente, dati i valori iniziali di $\underline{\boldsymbol{\lambda}}^k$ e ρ, finchè $\left\| L(\underline{\boldsymbol{x}}, \underline{\boldsymbol{\lambda}}^k, \rho) \right\| > \epsilon$ si procede ad ottenere il punto ottimo $\underline{\boldsymbol{x}}^*{}_k$ risolvendo $L(\underline{\boldsymbol{x}}, \underline{\boldsymbol{\lambda}}^k, \rho)$ rispetto a $\underline{\boldsymbol{x}}$ con un qualsiasi metodo di ottimizzazione svincolata, quindi si aggiorna il valore di λ come $\lambda_j^{k+1} = \lambda_k^k + 2\rho h_j(\underline{\boldsymbol{x}}^*{}_k)$. Eventualmente viene aggiornato anche il valore di ρ.

8.5 Sequential quadratic programming (SQP)

L'idea è di applicare il metodo di Newton per identificare $(\underline{\boldsymbol{x}}^*, \underline{\boldsymbol{\lambda}}^*)$ dalle condizioni KKT di un problema vincolato, riducendo ogni step del metodo di Newton in un problema di programmazione quadratica. Considerando un problema generale:

$$\begin{aligned} &\min f(\underline{\boldsymbol{x}}) \\ &\underline{\boldsymbol{g}}(\underline{\boldsymbol{x}}) \leqslant 0 \\ &\underline{\boldsymbol{h}}(\underline{\boldsymbol{x}}) = 0 \end{aligned}$$

Possiamo costruirne il modello lagrangiano:

$$L(\underline{\boldsymbol{x}}, \underline{\boldsymbol{\lambda}}, \underline{\boldsymbol{\mu}}) = f(\underline{\boldsymbol{x}}) + \sum_{j=1}^{k} \lambda_j g_j(\underline{\boldsymbol{x}}) + \sum_{j=1}^{h} \mu_j h_j(\underline{\boldsymbol{x}})$$

Data un'approssimazione della soluzione e dei moltiplicatori lagrangiani $\left(\underline{\boldsymbol{x}}_k, \underline{\boldsymbol{\lambda}}_k, \underline{\boldsymbol{\mu}}_k\right)$, nota la matrice hessiana di L possiamo scrivere:

$$\nabla^2 L(\underline{\boldsymbol{x}}_k) = H(\underline{\boldsymbol{x}}_k) + \sum_{j=1}^{k} \lambda_j^k \nabla^2 g_j(\underline{\boldsymbol{x}}_k) + \sum_{j=1}^{h} \mu_j^k \nabla^2 h_j(\underline{\boldsymbol{x}}_k)$$

Per aggiornare i valori dobbiamo identificare la direzione di Newton $\underline{\boldsymbol{d}}$ tramite il sotto-problema quadratico seguente:

$$\begin{aligned} \min \phi(\underline{\boldsymbol{x}}) = f(\underline{\boldsymbol{x}}_k) + \nabla f(\underline{\boldsymbol{x}}_k)^T \underline{\boldsymbol{d}} + \frac{1}{2} \underline{\boldsymbol{d}}^T \nabla^2 L(\underline{\boldsymbol{x}}_k) \underline{\boldsymbol{d}} \\ \underline{\boldsymbol{g}}(\underline{\boldsymbol{x}}_k) + \left[\frac{\partial \underline{\boldsymbol{g}}(\underline{\boldsymbol{x}}_k)}{\partial \underline{\boldsymbol{x}}} \right] \underline{\boldsymbol{d}} \leqslant 0 \\ \underline{\boldsymbol{h}}(\underline{\boldsymbol{x}}_k) + \left[\frac{\partial \underline{\boldsymbol{h}}(\underline{\boldsymbol{x}}_k)}{\partial \underline{\boldsymbol{x}}} \right] \underline{\boldsymbol{d}} = 0 \end{aligned}$$

Risolvendo il problema si ottiene $\underline{\boldsymbol{x}}_{k+1} = \underline{\boldsymbol{x}}_k + \underline{\boldsymbol{d}}_k$ ed i moltiplicatori lagrangiani aggiornati $\underline{\boldsymbol{\lambda}}_{k+1}$ e $\underline{\boldsymbol{\mu}}_{k+1}$. L'iterazione si interrompe basandosi su un criterio costruito sulla norma della direzione migliorante $\underline{\boldsymbol{d}}$.

Il Sequential quadratic programming identifica un punto che soddisfa le condizioni KKT, per cui tutti i punti non regolari sono evitati dall'algoritmo.

Parte II

Programmazione lineare intera

9

Algoritmi dei piani di taglio

Definizione 9.0.1 (Poliedro). Si definisce poliedro un insieme convesso del tipo:

$$\text{conv}(X) = \{\underline{\boldsymbol{x}} : \boldsymbol{A}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{b}}, \underline{\boldsymbol{x}} \geqslant 0\}$$

Definizione 9.0.2 (Disuguaglianza valida). Una disuguaglianza $\underline{\boldsymbol{\pi x}} \leqslant \underline{\boldsymbol{\pi}}_0$ è una disuguaglianza valida per $X \subseteq \mathbb{R}^n$ se $\underline{\boldsymbol{\pi x}} \leqslant \underline{\boldsymbol{\pi}}_0 \quad \forall \underline{\boldsymbol{x}} \in X$.

Proposizione 9.0.3 (Disuguaglianza valida per un problema). Una disuguaglianza $\underline{\boldsymbol{\pi x}} \leqslant \underline{\boldsymbol{\pi}}_0$ è valida per $P = \{\underline{\boldsymbol{x}} : \boldsymbol{A}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{b}}, \underline{\boldsymbol{x}} \geqslant 0\} \neq \emptyset$ se e solo se:

1. $\exists \underline{\boldsymbol{u}} \geqslant 0, \underline{\boldsymbol{v}} \geqslant 0 : \underline{\boldsymbol{u}}\boldsymbol{A} - \underline{\boldsymbol{v}} = \underline{\boldsymbol{\pi}} \quad \wedge \quad \underline{\boldsymbol{ub}} \leqslant \underline{\boldsymbol{\pi}}_0$, oppure
2. $\exists \underline{\boldsymbol{u}} \geqslant 0 : \underline{\boldsymbol{u}}\boldsymbol{A} \geqslant \underline{\boldsymbol{\pi}} \quad \wedge \underline{\boldsymbol{ub}} \leqslant \underline{\boldsymbol{\pi}}_0$

Proposizione 9.0.4 (Disuguaglianza valida per un problema intero). Sia $X = \left\{\underline{\boldsymbol{y}} \in \mathbb{Z}^1 : \underline{\boldsymbol{y}} \leqslant \underline{\boldsymbol{b}}\right\}$. Allora la diseguaglianza $\underline{\boldsymbol{y}} \leqslant \lfloor \underline{\boldsymbol{b}} \rfloor$ è valida per X.

9.1 Procedura di Chvatal-Gomory per la costruzione di disuguaglianze valide

Dato un set $X = P \cup \mathbb{Z}^n$, dove $P = \{\underline{\boldsymbol{x}} \in \mathbb{R}^n_+ : \boldsymbol{A}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{b}}\}$, $\boldsymbol{A}$ una matrice $n \times m$ e $\underline{\boldsymbol{u}} \in \mathbb{R}^m_+$, vale che:

La disuguaglianza $\sum_{j=1}^n ua_j x_j \leqslant \underline{\boldsymbol{ub}}$ è valida per $\boldsymbol{P}$ con $\underline{\boldsymbol{u}} \geqslant 0$ e $\sum_{j=1}^n a_j x_j \leqslant \underline{\boldsymbol{b}}$.

La disuguaglianza $\sum_{j=1}^n \lfloor ua_j \rfloor x_j \leqslant \underline{\boldsymbol{ub}}$ è valida per $\boldsymbol{P}$ con $\underline{\boldsymbol{x}} \geqslant 0$.

La disuguaglianza $\sum_{j=1}^n \lfloor ua_j \rfloor x_j \leqslant \underline{\boldsymbol{ub}}$ è valida per X con $\underline{\boldsymbol{x}} \in \mathbb{Z}^n_+$, e quindi $\sum_{j=1}^n \lfloor ua_j \rfloor x_j$ è un intero.

Teorema 9.1.1. Ogni disuguaglianza valida per X può essere ottenuta applicando la procedura di Chvatal-Gomory per un numero finito di volte.

9.2 Algoritmo dei piani di taglio

Un generico algoritmo dei piani di taglio per un problema intero del tipo $\max\{\underline{\boldsymbol{c}}\underline{\boldsymbol{x}} : \underline{\boldsymbol{x}} \in X\}$ che generi disuguaglianze utili da una certa famiglia $\mathscr{F}$ è della forma seguente:

9.2.1 Inizializzazione

Il termine dell'iterazione t viene inizializzato a zero ed il problema intero elaborato inizialmente coincide con quello originale: $P^0 = P$.

9.2.2 Iterazione

Si va a risolvere il seguente problema lineare:

$$\overline{z}^{(t)} = \max\{\underline{\boldsymbol{c}}\underline{\boldsymbol{x}} : \underline{\boldsymbol{x}} \in P^{(t)}\}$$

Sia quindi $\underline{\boldsymbol{x}}^{*(t)}$ la soluzione ottima ottenuta:

1. Se $\underline{\boldsymbol{x}}^{*(t)} \in \mathbb{Z}^n$ allora l'algoritmo si interrompe e $\underline{\boldsymbol{x}}^{*(t)}$ è una soluzione ottima della programmazione intera.
2. Se $\underline{\boldsymbol{x}}^{*(t)} \notin \mathbb{Z}^n$ si risolve il problema di separazione per $\underline{\boldsymbol{x}}^{*(t)}$ e $\mathscr{F}$.
   (a) Se viene identificata una disuguaglianza $\left(\underline{\boldsymbol{\pi}}^{(t)}, \underline{\boldsymbol{\pi}}_0^{(t)}\right) \in \mathscr{F} : \underline{\boldsymbol{\pi}}^{(t)}\underline{\boldsymbol{x}}^{*(t)} > \underline{\boldsymbol{\pi}}_0^{(t)}$ tale che viene tagliata la soluzione $\underline{\boldsymbol{x}}^{*(t)}$, allora essa viene aggiunta ai vincoli del problema: $P^{(t+1)} = P^{(t)} \cap \left\{\underline{\boldsymbol{x}} : \underline{\boldsymbol{\pi}}^{(i)}\underline{\boldsymbol{x}}^{*(i)} > \underline{\boldsymbol{\pi}}_0^{(i)}\right\}$.
   (b) Se non viene identificata, termina.

9.2.3 Considerazioni sul risultato

Se l'algoritmo termina senza aver identificato una soluzione intera, il problema risultante $P^{(t)}$ rimane comunque un buon punto di partenza per eseguire un algoritmo branch-and-bound.

9.3 Algoritmo dei piani di taglio frazionari di Gomory

Considerato un problema intero del tipo $\max\{\underline{\boldsymbol{c}}\underline{\boldsymbol{x}} : \boldsymbol{A}\underline{\boldsymbol{x}} = \underline{\boldsymbol{b}}, \underline{\boldsymbol{x}} \in \mathbb{N}^n\}$, si procede ad identificare una base ottima $\boldsymbol{B}$ del rilassamento continuo, si sceglie una variabile di base $x_{\boldsymbol{B}_u}$ che sia frazionaria e quindi si procede a generare una disuguaglianza di Chvatal-Gomory associata con questa variabile in modo tale da tagliare la soluzione dalla regione ammissibile, scegliendo la riga della matrice dei vincoli $\boldsymbol{A}$ tale per cui un coefficiente $\overline{a}_{uj}$ non sia intero (deve esistere, altrimenti la soluzione ottima sarebbe intera). Se la base fosse priva di variabili frazionarie ovviamente si avrebbe già identificato la soluzione ottima del problema intero.

$$x_{\boldsymbol{B}_u} + \sum_{i \in |\boldsymbol{B}|} \lfloor \overline{a}_{uj} \rfloor x_j \leqslant \lfloor \overline{a}_{u0} \rfloor$$

(a) Taglio di Chvatal-Gomory per la riga u

$$\sum_{i \in |\boldsymbol{B}|} (\overline{a}_{uj} - \lfloor \overline{a}_{uj} \rfloor) x_j \leqslant \overline{a}_{u0} - \lfloor \overline{a}_{u0} \rfloor$$

(b) Taglio di Chvatal-Gomory per la riga u eliminando la variabile $x_{\boldsymbol{B}_u}$

$$\sum_{i \in |\boldsymbol{B}|} f_{uj} x_j \leqslant f_{u0}$$

(c) Taglio di Chvatal-Gomory ponendo $f_{uj} = \overline{a}_{uj} - \lfloor \overline{a}_{uj} \rfloor$

Figura 9.1: Esempi di tagli di Chvatal-Gomory (equivalenti)

9.3.1 Variabile di slack

La variabile di slack s di un taglio di Chvatal-Gomory si ottiene come:

$$s = -f_{u0} + \sum_{j \in |\boldsymbol{B}|} f_{uj} x_j$$

Figura 9.2: Variabile di slack di un taglio di Chvatal-Gomory

Proposizione 9.3.1 (Taglio di Gomory e disuguaglianza di Chvatal-Gomory). Sia β la riga u-esima di $\boldsymbol{B}^{-1}$ e $q_i = \beta_i - \lfloor \beta_i \rfloor \; \forall i$. Il taglio di Gomory $\sum_{i \in |\boldsymbol{B}|} f_{uj} x_j \leqslant f_{u0}$, quando scritto in termini della variabile originale coincide con la disuguaglianza di Chvatal-Gomory:

$$\sum_{j=1}^{n} \lfloor q a_j \rfloor x_j \leqslant \lfloor q b \rfloor$$

9.4 Tagli interi misti

Proposizione 9.4.1 (Disuguaglianza intera mista semplice). Sia $X^{\geqslant} = \{(x,y) \in \mathbb{R}^1_+ \times \mathbb{Z}^1 : x + y \geqslant b\}$ e sia $f = b - \lfloor b \rfloor > 0$. La disuguaglianza:

$$x \geqslant f(\lceil b \rceil - y) \quad \vee \quad \frac{x}{f} + y \geqslant \lceil b \rceil$$

risulta valida per $X^{\geqslant}$

Corollario 9.4.1.1. Se $X^{\leqslant} = \{(x,y) \in \mathbb{R}^1_+ \times \mathbb{Z}^1 : y \leqslant b + x\}$ e sia $f = b - \lfloor b \rfloor > 0$, la disuguaglianza

$$y \leqslant \lfloor b \rfloor + \frac{x}{1-f}$$

È valida per $X^{\leqslant}$.

Proposizione 9.4.2 (Disuguaglianza intera arrotondata mista (MIR)). Questa variante della disuguaglianza semplice si ottiene considerando il set:

$$X^{MIR} = \{(x,y) \in \mathbb{R}^1_+ \times \mathbb{Z}^2_+ : a_1 y_1 + a_2 y_2 \leqslant b + x\}$$

Dove a_1, a_2 e b sono degli scalari con b frazionario.

Sia $f = b - \lfloor b \rfloor$ ed $f_i = a_i - \lfloor a_i \rfloor \quad i \in \{1,2\}$. Se $f_1 \leqslant f \leqslant f_2$, allora:

$$\lfloor a_1 \rfloor y_1 + \left(\lfloor a_2 \rfloor + \frac{f_2 - f}{1-f} \right) y_2 \leqslant \lfloor b \rfloor + \frac{x}{1-f}$$

È valida per X^{MIR}

Proposizione 9.4.3 (Taglio di Gomory intero misto). Dato un tableau di un problema intero, contenente una **variabile di base frazionaria** $y_{\boldsymbol{B}_u}$ che quindi genera un set del tipo:

$$X^{(G)} = \left\{ (y_{\boldsymbol{B}_u}, \underline{\boldsymbol{y}}, \underline{\boldsymbol{x}}) \in \mathbb{Z}^1 \times \mathbb{Z}^{n_1}_+ \times \mathbb{Z}^{n_2}_+ : y_{\boldsymbol{B}_u} + \sum_{j \in N_1} \overline{a}_{uj} y_j + \sum_{j \in N_2} \overline{a}_{uj} x_j = \overline{a}_{u0} \right\}$$

Se $\overline{a}_{u0} \notin \mathbb{Z}^1$, $f_j = \overline{a}_{uj} - \lfloor \overline{a}_{uj} \rfloor \; \forall j \in N_1 \cup N_2$ e $f_0 = \overline{a}_{u0} - \lfloor \overline{a}_{u0} \rfloor$ il taglio di Gomory misto:

$$\sum_{f_j \leqslant f_0} f_j y_j + \sum_{f_j > f_0} \frac{f_0(1-f_j)}{1-f_0} y_j + \sum_{\overline{a}_{uj} > 0} \overline{a}_{uj} x_j + \sum_{\overline{a}_{uj} < 0} \frac{f_0}{1-f_0} \overline{a}_{uj} x_j \geqslant f_0$$

È valida per X^G. La formula si ottiene calcolando la MRI per il set e sostituire la variabile $y_{\boldsymbol{B}_u}$ con la forma in f_j.

10 Disuguaglianze forti valide

Per disuguaglianze*forti* si intende una disuguaglianza che porta ad una formulazione *migliore*. Purtroppo, spesso, descrivere famiglie di disuguaglianze forti risulta piuttosto complesso ed il problema di separazione che vanno a porre altrettanto, che talvolta risulta essere **NP-hard**.

Queste disuguaglianze vengono usate per risolvere problemi grandi e/o complessi integrandole in un modello branch-and-bound, che nello specifico viene chiamato **branch-and-cut**.

10.1 Definizioni preliminari

10.1.1 Definizioni sulle disuguaglianze

Figura 10.1: Disuguaglianze equivalenti

Definizione 10.1.1 (Disuguaglianze equivalenti). Due disuguaglianze $\underline{\boldsymbol{\pi}}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{\pi}}_0$ e $\underline{\boldsymbol{\mu}}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{\mu}}_0$ valide per $P \subseteq \mathbb{R}^n_+$ sono **equivalenti** (10.1) se:

$$\exists \lambda > 0 : \underline{\boldsymbol{\pi}} = \lambda \underline{\boldsymbol{\mu}} \quad \wedge \underline{\boldsymbol{\pi}}_0 = \lambda \underline{\boldsymbol{\mu}}_0$$

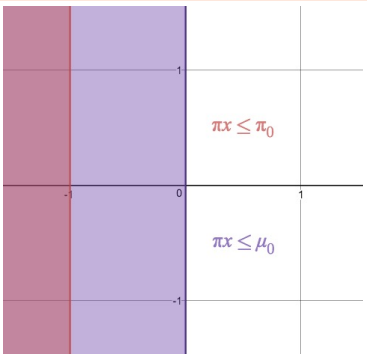

Figura 10.2: Disuguaglianze dominanti e ridondanti

Definizione 10.1.2 (Disuguaglianze dominanti). Se due disuguaglianze $\Pi : \underline{\boldsymbol{\pi}}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{\pi}}_0$ e $M : \underline{\boldsymbol{\mu}}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{\mu}}_0$ sono valide per $P \subseteq \mathbb{R}^n_+$, si dice che Π **domina** M 10.2 se esse non sono equivalenti e:

$$\exists \lambda > 0 : \underline{\boldsymbol{\pi}} \geqslant \lambda \mu \quad \underline{\boldsymbol{\pi}}_0 \leqslant \lambda \mu_0$$

Il poliedro descritto dalla disuguaglianza dominante è contenuto nel poliedro della disuguaglianza dominata.

Definizione 10.1.3 (Disuguaglianze ridondante). Una disuguaglianza valida $\Pi : \underline{\boldsymbol{\pi}\boldsymbol{x}} \leqslant \underline{\boldsymbol{\pi}}_0$ è detta **ridondante** nella descrizione di $\boldsymbol{P}$ se esistono una o più disuguaglianze valide su $\boldsymbol{P}$ che la dominano.

Una disuguaglianza non ridondante è detta **unica**.

Dove non è possibile ottenere una descrizione esplicita di $P = \text{conv}(X)$, come nel caso di problemi NP-hard (a meno che $\mathscr{P} = \mathscr{NP}$), identificare equazioni ridondanti può risultare molto difficile.

10.1.2 Definizioni sui poliedri

Definizione 10.1.4 (Poliedro dimensionalmente completo). Un poliedro $P \subseteq \mathbb{R}^n$ che contiene n direzioni linearmente indipendenti è detto **dimensionalmente completo**.

Proposizione 10.1.5 (Non-esistenza di equivalenze valide per poliedri dimensionalmente completi). Se $\boldsymbol{P}$ è un poliedro dimensionalmente completo, allora non esiste un'equazione $\underline{\boldsymbol{\alpha}\boldsymbol{x}} = \underline{\boldsymbol{b}}$ che sia valida $\forall \underline{\boldsymbol{x}} \in P$.

Teorema 10.1.6 (Descrizione minimale unica di un poliedro dimensionalmente completo). Se $\boldsymbol{P}$ è un poliedro dimensionalmente completo, esso possiede una descrizione minimale unica:

$$P = \left\{\underline{\boldsymbol{x}} \in \mathbb{R}^n : a_i \underline{\boldsymbol{x}} \leqslant b_i \quad \forall i \in [1, \dots, m]\right\}$$

dove ogni disuguaglianze è unica.

Definizione 10.1.7 (Punti indipendenti affini). Un insieme di punti $\{\underline{\boldsymbol{x}}_1, \dots, \underline{\boldsymbol{x}}_k\} \in \left\{\mathbb{R}^n\right\}^k$ sono detti **indipendenti affini** se le $k-1$ direzioni $\underline{\boldsymbol{x}}_2 - \underline{\boldsymbol{x}}_1, \dots, \underline{\boldsymbol{x}}_k - \underline{\boldsymbol{x}}_1$ sono linearmente indipendenti.

Definizione 10.1.8 (Dimensione di un poliedro). Con dimensione di un poliedro $P \subseteq \mathbb{R}^n$, è pari a $\dim(P) = \#\left\{\underline{\boldsymbol{x}}_{\text{ind. aff.}} \in P\right\} - 1$, cioè contiene n direzioni linearmente indipendenti.

Definizione 10.1.9 (Faccia di poliedro). F definisce una **faccia** di un poliedro $\boldsymbol{P}$ se $F = \left\{\underline{\boldsymbol{x}} \in P : \underline{\boldsymbol{\pi}\boldsymbol{x}} = \underline{\boldsymbol{\pi}}_0\right\}$ per una disuguaglianza valida $\underline{\boldsymbol{\pi}\boldsymbol{x}} = \underline{\boldsymbol{\pi}}_0$ di $\boldsymbol{P}$.

Una faccia di un poliedro è a sua volta un poliedro ed il numero di facce di un poliedro è finito.

Definizione 10.1.10 (Sfaccettatura di poliedro). F definisce una **sfaccettatura** di un poliedro $\boldsymbol{P}$ se è una faccia di $\boldsymbol{P}$ e $\dim(F) = \dim(P) - 1$.

Definizione 10.1.11 (Disuguaglianze rappresentante (o definente) una faccia di poliedro). Data una **faccia** F di un poliedro $\boldsymbol{P}$ definita come $F = \left\{\underline{\boldsymbol{x}} \in P : \underline{\boldsymbol{\pi}\boldsymbol{x}} = \underline{\boldsymbol{\pi}}_0\right\}$, la disuguaglianza che la caratterizza è detta **rappresentante** la faccia.

Proposizione 10.1.12 (Disuguaglianza necessaria). Se il poliedro $\boldsymbol{P}$ è dimensionalmente completo, una disuguaglianza valida $\underline{\boldsymbol{\pi}\boldsymbol{x}} \leqslant \underline{\boldsymbol{\pi}}_0$ è **necessaria** nella descrizione di $\boldsymbol{P}$ se e solo se rappresenta una **sfaccettatura** di $\boldsymbol{P}$.

Ciò significa che una disuguaglianza definisce una sfaccettatura di $\boldsymbol{P}$ se e solo se esistono n punti indipendenti $\underline{\boldsymbol{x}} \in P$ che ne soddisfano la forma in uguaglianza: questi punti saranno quindi posizionati su un generico iperpiano $\underline{\boldsymbol{\mu}\boldsymbol{x}} = \underline{\boldsymbol{\mu}}_0$.

$$\sum_{j=1}^{n} \mu_j \underline{\boldsymbol{x}}_{jk} = \mu_0 \quad \forall k$$

Se l'unica soluzione ammissibile è per $(\mu, \mu_0) = \lambda(\pi, \pi_0) \quad \lambda \neq 0$ allora $\underline{\boldsymbol{\pi}\boldsymbol{x}} \leqslant \underline{\boldsymbol{\pi}}_0$ è rappresenta una sfaccettatura ed è quindi **necessaria**.

10.2 Dimostrare che un poliedro descrive un insieme convesso

Esistono diversi approcci per mostrare che un poliedro $P = \{\underline{\boldsymbol{x}} \in \mathbb{R}^n : \boldsymbol{A}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{b}}\}$ descrive conv(X):

1. Mostrare che la matrice $\boldsymbol{A}$, o la coppia $\boldsymbol{A}, \underline{\boldsymbol{b}}$ ha una struttura particolare che garantisce che $P = \text{conv}(X)$.
2. Mostrare che i punti $\left\{\underline{\boldsymbol{x}}, \underline{\boldsymbol{y}}\right\} \in P$ con $\underline{\boldsymbol{y}}$ frazionario non sono **vertici** di $\boldsymbol{P}$.
3. Mostrare che per qualsiasi punto $\underline{\boldsymbol{c}} \in \mathbb{R}^n$, il programma lineare $z_{LP} = \max\{\underline{\boldsymbol{cx}} : \boldsymbol{A}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{b}}\}$ ha una soluzione ottima $\underline{\boldsymbol{x}}^* \in X$.
4. Mostrare che $\forall \underline{\boldsymbol{c}} \in \mathbb{R}^n$ esiste un punto $\underline{\boldsymbol{x}}^* \in X$ e una soluzione ammissibile $\underline{\boldsymbol{u}}^*$ del problema duale $\omega_{LP} = \min\{\underline{\boldsymbol{ub}} : \underline{\boldsymbol{u}}\boldsymbol{A} = \underline{\boldsymbol{c}}, \underline{\boldsymbol{u}} \geqslant 0\}$ con $\underline{\boldsymbol{cx}}^* = \underline{\boldsymbol{u}}^* \underline{\boldsymbol{b}}$.
5. Mostrare che se $\underline{\boldsymbol{\pi x}} \leqslant \underline{\boldsymbol{\pi}}_0$ rappresenta una sfaccettatura di conv(X), allora deve essere identica a una delle disuguaglianze che rappresentano $\boldsymbol{P}$.
6. Mostrare che $\forall \underline{\boldsymbol{c}} \in \mathbb{R}^n, \underline{\boldsymbol{c}} \neq 0$ l'insieme delle soluzioni ottime di $\max\{\underline{\boldsymbol{cx}} : \underline{\boldsymbol{x}} \in X\}$ è contenuto in una delle disuguaglianze rappresentanti $\boldsymbol{P}$.
7. Verificare che $\underline{\boldsymbol{b}} \in \mathbb{Z}^n$ e che $\forall \underline{\boldsymbol{c}} \in \mathbb{Z}^n$, il valore ottimo del problema duale ω_{LP} ha valore intero.

10.3 Cover Inequalities

Consideriamo il set di possibili soluzioni di un problema dello zaino a pesi e peso massimo strettamente positivi $X = \{\underline{x} \in B^n : \sum_{i=1}^{n} \underline{x}_i a_i \leqslant b\}$.

Definizione 10.3.1 (Cover). Un set $S \subseteq N$ è una **cover** se $\sum_{i \in S} a_i > b$. Una cover è detta **minimale** se $C \setminus \{j\}$ non è una cover per nessun $j \in C$.

Il **vettore di incidenza associato** alla cover $\underline{x}^C$ **non appartiene** alla regione ammissibile definita da X.

Se un set $T \subseteq N$ contiene una cover C, allora anche T è una cover.

Proposizione 10.3.2 (Validità di una cover inequality). Se $C \subseteq N$ è una cover per X, allora la cover inequality

$$\sum_{j \in C} x_j \leqslant |C| - 1$$

È valida per X

10.4 Irrobustire una cover inequality

Proposizione 10.4.1. Se C è una cover per X, la cover inequality estesa

$$\sum_{j \in E(C)} x_j \leqslant |C| - 1$$

è valida per X, dove $E(C) = C \cup \{j : a_j \geqslant a_i \quad \forall i \in C\}$

10.5 Sollevare le cover inequalities

In generale il problema risulta essere trovare il miglior valore possibile di un peso $\alpha_j, j \in N\backslash C$ in modo tale che la disuguaglianza

$$\sum_{j\in C} x_j + \sum_{j\in N\backslash C} \alpha_j \underline{\boldsymbol{x}}_j \leqslant |C| - 1$$

sia valida in X.

La procedura di sollevamento delle cover inequalities è un metodo per produrre questo set di valori.

Siano $j_1, \ldots, j_r$ un insieme ordinato di $N\backslash C$ e sia t uno scalare inizialmente di valore pari a 1. Si inizia dalla migliore uguaglianza identificata sino ad ora:

$$\sum_{j\in C} x_j + \sum_{i=1}^{t-1} \alpha_{j_i} \underline{\boldsymbol{x}}_{j_i} \leqslant |C| - 1$$

E si procede ad identificare il massimo valore di α_{j_i} per cui essa risulta essere valida risolvendo il seguente problema dello zaino:

$$\begin{aligned} \xi_t = \max & \sum_{j\in C} x_j + \sum_{i=1}^{t-1} \alpha_{j_i} \underline{\boldsymbol{x}}_{j_i} \\ & \sum_{j\in C} x_j + \sum_{i=1}^{t-1} a_{j_i} \underline{\boldsymbol{x}}_{j_i} \leqslant b - a_{j_t} \\ & x \in \{0,1\}^{|C|+t-1} \end{aligned}$$

Figura 10.3: Problema dello zaino per il sollevamento delle cover inequalities

Si inizializza $\alpha_{j_t} = |C| - 1 - \xi_t$ e si interrompe l'iterazione quando $t = r$.

10.6 Separazione delle cover inequalities

Sia $\mathscr{F}$ la famiglia delle cover inequalities per X e procediamo ad esaminare il problema della separazione su questa famiglia. Dato un punto non-intero $\underline{\boldsymbol{x}}^* \in [0,1]^n$ vogliamo verificare se esso soddisfi o meno tutte le cover inequalities. Questo può essere formalizzato come:

$$\sum_{j\in C} (1 - x_j) \geqslant 1$$

La domanda che ci poniamo quindi è la seguente: esiste un set $C \subseteq N$ tale che $\sum_{j\in X} a_j > b$ per cui $\sum_{j\in C}(1 - x_j^*) < 1$?

Espresso in termini formali, la seguente disequazione è valida?

$$\xi = \min \left\{ \sum_{j\in N} (1 - x_j^*) z_j : \sum_{j\in N} a_j z_j > b, z \in B^n \right\} < 1$$

Teorema 10.6.1. 1. Se $\xi \geqslant 1$ allora x^* soddisfa tutte le cover inequalities.

2. Se $\xi \leqslant 1$ con come soluzione ottimale z^R, allora la cover inequality $\sum_{j\in R} x_j \leqslant |R| - 1$ taglia il punto $\underline{\boldsymbol{x}}^*$ di un valore pari $1 - \xi$.

10.7 Disuguaglianze cover per i flussi

Definizione 10.7.1 (Cover generalizzata). Un insieme $C = C_1 \cup C_2$ con $C_1 \subseteq N_1, C_2 \subseteq N_2$ è una **cover generalizzata** per X se:

$$\sum_{j \in C_1} a_j - \sum_{j \in C_2} a_j = b + \lambda, \quad \lambda > 0$$

Il termine λ è chiamato **eccesso della cover**

Proposizione 10.7.2. La disuguaglianza cover di flusso:

$$\sum_{j \in C_1} x_j + \sum_{j \in C_1} (a_j - \lambda)^+ (1 - y_j) \leqslant b + \sum_{j \in C_2} a_j + \lambda \sum_{j \in L_2} y_j + \sum_{j \in N_2 \setminus \{C_2 \cup L_2\}} x_j$$

È valida per X, dove $L_2 \subseteq N_2 \setminus C_2$.

10.7.1 Separazione per le disuguaglianze cover per i flussi

Se si procede con l'assunzione che $x_j = a_j y_j$, $a_j \geqslant \lambda \quad \forall j \in C_1$ e che $L_2 = N_2 \setminus C_2$, la disuguaglianza cover per i flussi diviene:

$$\begin{aligned}
\sum_{j \in C_1} a_j y_j + \sum_{j \in C_1} (a_j - \lambda)^+ (1 - y_j) &\leqslant b + \sum_{j \in C_2} a_j + \lambda \sum_{j \in L_2} y_j + \sum_{j \in N_2 \setminus \{C_2 \cup L_2\}} a_j y_j \\
\sum_{j \in C_1} a_j - \lambda \sum_{j \in C_1} (1 - y_j) &\leqslant b + \sum_{j \in C_2} a_j + \lambda \sum_{j \in N_2 \setminus C_2} y_j \\
\sum_{j \in C_1} (1 - y_j) + \sum_{j \in N_2 \setminus C_2} y_j &\geqslant 1 \\
\sum_{j \in C_1} (1 - y_j) - \sum_{j \in C_2} y_j &\geqslant 1 - \sum_{j \in N_2} y_j
\end{aligned}$$

Questo risultato coincide con le disuguaglianze cover per il problema dello zaino con pesi positivi e negativi:

$$\left\{ \underline{x} \in B^n : \sum_{j \in N_1} a_j y_j - \sum_{j \in N_2} a_j y_j \leqslant b \right\}$$

Figura 10.4: Problema dello zaino con pesi positivi e negativi

Dove la cover (C_1, C_2) deve soddisfare $\sum_{j \in C_1} a_j - \sum_{j \in C_2} a_j \leqslant b + \lambda \quad \lambda > 0$.

Da questo risulta possibile costruire un'euristica di separazione per le disuguaglianze di cover per i flussi: se z è il vettore di incidenza della cover $C = (C_1, C_2)$:

$$\begin{aligned}
\xi = \min \sum_{j=N_1} (1 - y_j^*) z_j - \sum_{j=N_2} (1 - y_j^*) z_j \\
\sum_{j \in N_1} a_j z_j - \sum_{j \in N_2} a_j z_j > b \\
\underline{z} \in 0, 1^n
\end{aligned}$$

Figura 10.5: Problema dello zaino per ottenere il vettore di incidenza della cover

Data la cover $C = (C_1, C_2)$ ottenuta risolvendo il problema dello zaino 10.5, basta verificare se:

$$\sum_{j \in C_1} x_j^* + \sum_{j \in C_1} (a_j - \lambda) + \left(1 - y_j^*\right) > b + \sum_{j \in C_2} a_j + \lambda \sum_{j \in L_2} y_j^* + \sum_{j \in N_2 \setminus \{C_2 \cup L_2\}} x_j^*$$

dove $L_2 \subseteq N_2 \setminus C_2$. Se la disuguaglianza è vera, allora è stata identificata una disuguaglianza cover violata.

10.8 Branch-and-cut

Un algoritmo **branch-and-cut** è un tipo di **branch-and-bound** in cui i piani di taglio sono generati tramite l'albero di esplorazione. Ad ogni nodo non solo si cerca di identificare il vincolo più stringente e migliorare la formulazione per ridurre il numero dei nodi dell'albero, ma si procede anche a ripetere una fase di proprocessing ed euristiche ad ogni nodo.

L'algoritmo procede nel modo seguente:

10.8.1 Inizializzazione

Si inizia da un problema $z = \max\{\underline{\boldsymbol{c}}\underline{\boldsymbol{x}} : \underline{\boldsymbol{x}} \in X\}$ con una formulazione $\boldsymbol{P}$. Inizializziamo $\overline{z} = -\infty, \underline{\boldsymbol{x}}^* = \emptyset$, si preprocessa il problema iniziale z e viene messo nella lista dei nodi.

10.8.2 Procedura

1. Se la lista dei nodi è vuota, l'algoritmo termina.
2. Altrimenti viene estratto un nodo i dalla lista, rappresentante la formulazione P^i dell'insieme X^i e si inizializza $P^{i,1} = P^i$.
   (a) Si risolve il problema rilassato $\overline{z}^{i,k} = \max\{\underline{\boldsymbol{c}}\underline{\boldsymbol{x}} : \underline{\boldsymbol{x}} \in P^{i,k}\}$. Se è impossibile, si ritorna al passo 1.
   (b) Altrimenti si procede ad identificare un taglio per eliminare la soluzione $\underline{\boldsymbol{x}}^{i,k}$ ed identificare una formulazione $P^{i,k+1} \subseteq P^{i,k}$ e si ripete il passo 2a.
   (c) Se nessun ulteriore taglio è identificato, si confronta il valore della soluzione identificata $\overline{z}^{i,k}$ con quella ottima. Se è peggiore o uguale si riparte dal passo 1.
   (d) Se è migliore e $\underline{\boldsymbol{x}}^{i,k}$ è ammissibile nel problema intero, si aggiorna $\underline{\boldsymbol{x}}^* = \underline{\boldsymbol{x}}^{i,k}, \overline{z} = \overline{z}^{i,k}$ e si riparte dal passo 1.
   (e) Se è migliore ma $\underline{\boldsymbol{x}}^{i,k}$ non è ammissibile nel problema intero, vanno create due o più nuovi problemi $X^{(it)}$ e le rispettive formulazioni $P^{(it)}$ e vanno aggiunte alla lista. Si riparte quindi dal passo 1.

11 Dualità lagrangiana

11.1 Rilassamento lagrangiano

Viene utilizzata per risolvere problemi di programmazione intera. Dato un problema con vincoli complessi:

$$\begin{aligned} z = \max \underline{\boldsymbol{c}}^T \underline{\boldsymbol{x}} \\ \boldsymbol{D}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{b}} \\ \underline{\boldsymbol{x}} \in X \end{aligned}$$

Il seguente è il corrispendente **rilassamento lagrangiano**:

Definizione 11.1.1 (Rilassamento lagrangiano). Dato un vettore di moltiplicatori di Lagrange $\underline{\boldsymbol{\lambda}} \in \mathbb{R}^m, \underline{\boldsymbol{\lambda}} \geqslant 0$, si definisce **rilassamento lagrangiano** del problema $\boldsymbol{P}$ il seguente problema di ottimizzazione:

$$\begin{aligned} z = \max \underline{\boldsymbol{c}}^T \underline{\boldsymbol{x}} \\ \boldsymbol{D}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{b}} \\ \underline{\boldsymbol{x}} \in X \end{aligned}$$

(a) Problema $\boldsymbol{P}$

$$\begin{aligned} z_{RL} = \max \underline{\boldsymbol{c}}^T \underline{\boldsymbol{x}} - \underline{\boldsymbol{\lambda}}^T (\boldsymbol{D}\underline{\boldsymbol{x}} - \underline{\boldsymbol{b}}) \\ \boldsymbol{C}\underline{\boldsymbol{x}} \leqslant \underline{\boldsymbol{d}} \\ \underline{\boldsymbol{x}} \in X \end{aligned}$$

(b) Rilassamento lagrangiano di $\boldsymbol{P}$

Figura 11.1: Il rilassamento lagrangiano

Teorema 11.1.2 (Condizioni di integralità). Dato un problema di PLI $\boldsymbol{P}$, il suo rilassamento continuo RC ed il suo rilassamento lagrangiano RL, se questi ammettono soluzione ottima, per qualsiasi vettore dei costi della funzione obbiettivo di $\boldsymbol{P}$ se:

$$\text{Conv}\left\{\underline{\boldsymbol{x}} \in \mathbb{Z}^n : \boldsymbol{C}\underline{\boldsymbol{x}} \geqslant \underline{\boldsymbol{d}}, \underline{\boldsymbol{d}} \geqslant \underline{\boldsymbol{0}}\right\} = \left\{\underline{\boldsymbol{x}} \in \mathbb{R}^n : \boldsymbol{C}\underline{\boldsymbol{x}} \geqslant \underline{\boldsymbol{d}}, \underline{\boldsymbol{d}} \geqslant \underline{\boldsymbol{0}}\right\}$$

Allora $\underline{\boldsymbol{z}}^*{}_{RL} = \underline{\boldsymbol{z}}^*{}_{RC}$, cioè il rilassamento lagrangiano non può ottenere una limitazione inferiore sul valore ottimo migliore di quella ottenibile dal rilassamento lineare.

Teorema 11.1.3 (Condizioni di dualità lagrangiana debole). Sia $\overline{\underline{\boldsymbol{x}}}$ una soluzione ammissibile di $\boldsymbol{P}$, avente valore $\overline{z}_P$. Vale che:

$$\overline{z}_P \geqslant \overline{z}^*_{RL} \quad \forall \underline{\boldsymbol{\lambda}} \in \mathbb{R}^m, \lambda \geqslant 0$$

Teorema 11.1.4 (Condizioni di ortogonalità lagrangiana). Sia dato un problema di programmazione intera $\boldsymbol{P}$ ed il suo rilassamento lagrangiano RL, in corrispondenza di un dato vettore di moltiplicatori di Lagrange $\underline{\boldsymbol{\lambda}}^* \geqslant 0$. Sia $\underline{\boldsymbol{x}}^*{}_{RL}$ la soluzione ottima di RL. Se vale che:

1. $\underline{\boldsymbol{x}}^*{}_{RL}$ è ammissibile per $\boldsymbol{P}$, cioè $\boldsymbol{A}\underline{\boldsymbol{x}}^*{}_{RL} \geqslant \underline{\boldsymbol{b}}$
2. $\underline{\boldsymbol{\lambda}}^{*T}\left(\boldsymbol{A}\underline{\boldsymbol{x}}^*{}_{RL} - \underline{\boldsymbol{b}}\right) = 0$

Allora la soluzione $\underline{\boldsymbol{x}}^*{}_{RL}$ è ottima anche per $\boldsymbol{P}$.

Corollario 11.1.4.1 (Condizioni di ortogonalità lagrangiana per vincoli di uguaglianza). In questo caso se la soluzione ottima del rilassamento lagrangiano è ammissibile nel problema originario allora essa risulta anche essere ottima.

11.2 Risolvere il lagrangiano duale

In generale la funzione lagrangiana risulta non essere differenziabile (nei punti in cui cambia la pendenza). Si utilizza quindi al posto di un **gradiente** un **sottogradiente**.

Definizione 11.2.1 (Sottogradiente). Data una funzione concava $f(\underline{\boldsymbol{\lambda}}) : \mathbb{R}^m \to \mathbb{R}$ un vettore $s(\underline{\boldsymbol{\lambda}}') \in \mathbb{R}^m$ tale che:

$$f(\underline{\boldsymbol{\lambda}}) \leqslant f(\underline{\boldsymbol{\lambda}}') + s(\underline{\boldsymbol{\lambda}}')^T(\underline{\boldsymbol{\lambda}} - \underline{\boldsymbol{\lambda}}')\forall \lambda \in \mathbb{R}^m$$

viene detto **sottogradiente** di f in $\underline{\boldsymbol{\lambda}}' \in \mathbb{R}^m$.

Teorema 11.2.2 (Sottogradiente nel rilassamento lagrangiano). Siano $\underline{\boldsymbol{\lambda}}^{(t)} \in \mathbb{R}^m$ e $\underline{\boldsymbol{x}}^{*(t)}_{RL_{\underline{\boldsymbol{\lambda}}^{(t)}}} \in \mathbb{Z}^n$ due vettori che soddisfano la seguente condizione:

$$L^*(\underline{\boldsymbol{\lambda}}^{(t)}) = \underline{\boldsymbol{c}}^T\underline{\boldsymbol{x}}^{(t)} - \underline{\boldsymbol{\lambda}}^{(t)T}(\boldsymbol{A}\underline{\boldsymbol{x}}^{(t)} - \underline{\boldsymbol{b}}) = \min_{\underline{\boldsymbol{x}}}\left\{\underline{\boldsymbol{c}}^T\underline{\boldsymbol{x}} - \underline{\boldsymbol{\lambda}}^{(t)T}(\boldsymbol{A}\underline{\boldsymbol{x}} - \underline{\boldsymbol{b}}) : \boldsymbol{C}\underline{\boldsymbol{x}} \geqslant \underline{\boldsymbol{d}}; \underline{\boldsymbol{x}} \geqslant 0; \underline{\boldsymbol{x}} \in \mathbb{Z}^n\right\}$$

Cioè $\underline{\boldsymbol{x}}^{*(t)}_{RL_{\underline{\boldsymbol{\lambda}}^{(t)}}}$ è la soluzione ottima del lagrangiano con vettore di coefficienti $\underline{\boldsymbol{\lambda}}^{(t)}$. Allora il vettore $s(\underline{\boldsymbol{\lambda}}^{(t)}) = -(\boldsymbol{A}\underline{\boldsymbol{x}}^{(t)} - \underline{\boldsymbol{b}})$ è un **sottogradiente** di $L^*(\underline{\boldsymbol{\lambda}})$ in $\underline{\boldsymbol{\lambda}}^{(t)}$.

12

TSP simmetrico

Prendiamo in considerazione il problema del commesso viaggiatore su un grafo non orientato $\mathcal{G} = (N, E)$:

$\min \sum_{e \in E} c_e x_e$	Il costo degli archi inseriti deve essere minimo
$\sum_{e \in \delta(i)} x_e = 2 \quad \forall i$	Si deve entrare ed uscire da ogni nodo esattamente una volta
$\sum_{e \in E(S)} x_e \leqslant \lvert S \rvert - 1 \quad \forall S \subset N; 2 \leqslant \lvert S \rvert \leqslant \lvert N \rvert - 1$	Non possono esistere sotto-cicli, cioè in ogni sotto-insieme di n nodi vengono percorsi al $n-1$ archi
$x_e \in 0,1 \quad \forall e \in E$	Ogni arco può essere percorso o non percorso.

Figura 12.1: Modello TSP

È possibile riscrivere il vincolo dell'assenza di sotto-cicli (SEC, sub-tour elimination constraints) come il fatto che da ogni sotto-insieme di nodi entrino ed escano almeno due archi:

$\min \sum_{e \in E} c_e x_e$	Il costo degli archi inseriti deve essere minimo
$\sum_{e \in \delta(i)} x_e = 2 \quad \forall i$	Si deve entrare ed uscire da ogni nodo esattamente una volta
$\sum_{e \in \delta(S)} x_e \geqslant 2 \quad \forall S \subset N; 2 \leqslant \lvert S \rvert \leqslant \lvert N \rvert - 1$	Non possono esistere sotto-cicli, cioè in ogni sotto-insieme di nodi devono entrare ed uscire almeno due archi.
$x_e \in 0,1 \quad \forall e \in E$	Ogni arco può essere percorso o non percorso.

Figura 12.2: Modello TSP

12.1 Problema di separazione

I vincoli SEC sono in numero esponenziale e quindi si procede risolvendo il modello ignorandoli. Data quindi una soluzione $\underline{\boldsymbol{x}}^*$, se essa non definisce un **ciclo hamiltoniano**, si cerca di individuare un vincolo SEC violato, cioè si identifica un sotto insieme S per cui la disequazione $\sum_{e \in \delta(S)} x_e \geqslant 2$ risulta falsa.

Costruito un grafo composto dagli archi che sono stati scelti in $\underline{\boldsymbol{x}}^*$, il problema di separazione coincide ad identificare un taglio di capacità < 2 che coincide con il taglio di capacità minima. Quest'ultimo problema coincide, a sua volta, con il problema di calcolare il flusso massimo, che però è definito su un grafo orientato.

È possibile però evitare questa trasformazione utilizzando l'albero di Gomory-Hu.

12.2 Albero di Gomory-Hu

Si tratta di un albero ricoprente (N,T) degli $\mathscr{G}$ nodi di $\mathscr{G}' = (N,E^*)$. I lati che lo compongono non devono essere necessariamente essere i lati scelti in $\underline{\boldsymbol{x}}^*$. Dato un lato e, la sua rimozione crea due componenti connesse con insiemi di vertici $S, N\backslash S$. Ad ognuno di questi lati e è associato un peso $w(e)$ tale che:

1. Per ogni coppia di nodi, il valore di costo minimo che li separa in $\mathscr{G}$ è quello dell'unico cammino che li congiunge nell'albero, cioè il peso del taglio minimo dei due nodi nell'albero.
2. Per ogni lato $e$dell'albero, il peso $w(e)$ è quello del taglio associato ad e in $\mathscr{G}'$.

Per trovate il taglio di peso minimo in $\mathscr{G}'$ risulta sufficiente cercare il lato e di peso minimo nell'albero T.

12.3 Comb inequalities (Disuguaglianze a pettine)

Si tratta di una famiglia di diseguaglianze usate nella risoluzione del TSP simmetrico. Viene definita considerando un insieme di nodi H, il manico, e T, i denti, tali che:

$$
\begin{aligned}
H, T_1, T_2, \ldots, T_t &\subseteq N \\
T_j \setminus H &\neq \emptyset \forall\, j \\
T_j \cap H &\neq \emptyset \forall\, j \\
T_j \cap T_i &= \emptyset \forall\, j, i : i \neq j \\
t \geqslant 3 \quad &\wedge \quad \text{dispari}
\end{aligned}
$$

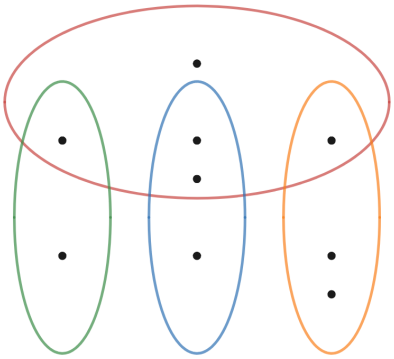

Figura 12.3: Comb Inequalities

La disuguaglianza afferma che:

$$x(E(H)) + \sum_{j=1}^{t} x(E(T_j)) \leqslant |H| + \sum_{j=1}^{t} \left|T_j\right| - \frac{3t+1}{2}$$

Dove $x(E(S)) = \sum_{e \in E(S)} x_e$ è l'insieme dei lati in un ciclo hamiltoniano i cui vertici appartengono entrambi all'insieme di vertici $S \subseteq N$. È possibile ricavare tale relazione a partire dai vincoli di grado:

$$2x(E(S)) + x(\delta(S)) = 2|S|$$

Dove $x(\delta(S)) = \sum_{e \in \delta(S)} x_e$ è l'insieme di lati in un ciclo hamiltoniano che hanno un estremo in S ed uno in $N \setminus S$.

Riscriviamo la relazione in funzione di x ed otteniamo $x(E(S)) = |S| - \frac{x(\delta(S))}{2}$.

Sommiamo tale relazione sugli insiemi $H, T_1, T_2, \ldots, T_t$ ed otteniamo:

$$x(E(H)) + \sum_{j=1}^{t} x(E(T_j)) = |H| + \sum_{j=1}^{t} \left|T_j\right| - \frac{1}{2}\left(x(\delta(H)) + \sum_{j=1}^{t} x(\delta(T_j))\right)$$

La disuguaglianza a pettine risulta vera se ricaviamo che $x(\delta(H)) + \sum_{j=1}^{t} x(\delta(T_j)) \geqslant 3t+1$.

Per farlo procediamo indicando con $x(\delta_j(H))$ l'insieme dei lati di un ciclo hamiltoniano con un estremo in $H \cap T_j$ ed uno non in H. Di conseguenza $x(\delta(H)) \geqslant \sum_{j=1}^{t} x(\delta_j(H))$ (il primo termine risulta maggiore perchp vi considero anche i lati di un ciclo hamiltoniano incidenti a vertici in H ma non nell'intersezione).

Per costruzione del ciclo che può toccare i vertici in $H \cap T_j$ e $T_j \setminus H$ risulta che $x(\delta_j(H)) + x(\delta(T_j)) \geqslant 3$.

Unendo le ultime due disuguaglianze si ottiene:

$$x(\delta(H)) + \sum_{j=1}^{t} x(\delta(T_j)) \geqslant \sum_{j=1}^{t} x(\delta_j(T)) + \sum_{j=1}^{t} x(\delta(T_j)) \geqslant 3t$$

È noto dalla prima relazione di uguaglianza che $x(\delta(S))$ sia un valore pari $\forall S$, mentre t è per costruzione dispari. Risulta sufficiente aggiungere 1 alla destra della disequazione per ricavare la relazione cercata:

$$x(\delta(H)) + \sum_{j=1}^{t} x(\delta(T_j)) \geqslant 3t+1$$

13

Tecnica di generazione delle colonne

Questo problema nasce dalla neccessità di fornire una tecnica di risoluzione a problemi dal numero di soluzioni esponenziali, che spesso risultano equivalenti: questo infatti manda in crisi gli elaboratori, che seguendo l'albero di ricerca devono considerare un numero esponenziale di soluzioni equivalenti. Il problema dei **cutting stock** ne è un esempio tipico.

13.1 Cutting stock

In questo problema è necessario tagliare delle assi (tutte di pari lunghezza) in modo da minimizzare il numero di assi utilizzate e soddisfare la richiesta totale.

13.1.1 Formulazione classica

$$\min \sum_{i=1}^{m} y_i$$ Va minimizzato il numero di assi che vengono utilizzate.

$$\sum_{j=1}^{n} l_j x_{ij} \leqslant L y_i \quad \forall i$$ Da ogni assa i si può realizzare al massimo un numero di prodotti la cui lunghezza totale non superi la lunghezza L dell'assa

$$\sum_{i=1}^{m} x_{ij} \geqslant d_j \quad \forall j$$ Bisogna realizzare almeno un numero di prodotti per tipo pari alla domanda.

$$x_{ij} \in \mathbb{Z}^+$$

$$y_i \in \{0,1\}$$

Figura 13.1: Cutting stock in formulazione classica

Il modello presenta un elevato grado di simmetria, cioè esistono tante soluzioni ottime equivalenti. Questo, come detto inizialmente, mette in crisi i risolutori ad albero.

13.1.2 Formulazione estesa

Consideriamo ora una formulazione che considera come variabili z_h i possibili schemi di taglio (considerando unicamente gli schemi non dominati, cioè che garantiscono il massimo numero di prodotti dallo stesso numero di tavole). Questo schema viene rappresentato da un vettore colonna $\underline{\boldsymbol{\alpha}} A_h = \begin{bmatrix} a_{1h} \\ \vdots \\ a_{nh} \end{bmatrix}$, in cui ogni elemento a_{jh} rappresenta il numero di prodotti j ricavati dallo schema di taglio h.

$$\min \sum_{h=1}^{H} z_h$$

Vogliamo minimizzare il numero di assi utilizzate: ogni schema di taglio si applica ad una sola asse.

$$\sum_{h=1}^{H} a_{jh} z_h \geqslant d_j$$

Il numero di prodotti realizzati deve rispondere alla domanda richiesta.

$$z_h \in \{0,1\} \quad \forall h$$

Figura 13.2: Cutting stock in formulazione estesa

Il numero $\boldsymbol{H}$ di schemi di taglio è solitamente esponenziale, per cui non è possibile elencare tutti i possibili z_h o le rispettive colonne A_h, per cui è necessaria adottare una tecnica per **generare** queste **colonne** in modo dinamico per poter arrivare a risolvere il rilassamento continuo del modello, che viene chiamato **Master Problem**.

13.2 Master Problem

Il master problem è il rilassamento continuo della formulazione estesa.

$$\min \sum_{h=1}^{H} z_h$$

Vogliamo minimizzare il numero di assi utilizzate: ogni schema di taglio si applica ad una sola asse.

$$\sum_{h=1}^{H} a_{jh} z_h \geqslant d_j$$

Il numero di prodotti realizzati deve rispondere alla domanda richiesta.

$$z_h \in [0,1] \quad \forall h$$

Figura 13.3: Master Problem

13.3 Master Problem ridotto

Il master problem **ridotto** è una versione del master problem realizzata considerando un numero piccolo di colonne di $\underline{\boldsymbol{A}}$, scelte in modo tale da continuare a rispettare i vincoli della domanda. Questo insieme ridotto viene indicato con $\overline{\underline{\boldsymbol{A}}}$.

$$\min \sum_{h=1}^{\overline{H}} z_h$$

Vogliamo minimizzare il numero di assi utilizzate: ogni schema di taglio si applica ad una sola asse.

$$\sum_{h=1}^{\overline{H}} a_{jh} z_h \geqslant d_j$$

Il numero di prodotti realizzati deve rispondere alla domanda richiesta.

$$z_h \in [0,1] \quad \forall h$$

Figura 13.4: Master Problem ridotto

13.4 Problema di Pricing

Una volta identificata una soluzione ottima z^* nel master problem ridotto è necessario verificare se essa risulti o meno ottima del master problem. Questo viene svolto cercando un **coefficiente di costo ridotto negativo** ottenuto tramite $\overline{c}_h = c_h - \underline{\boldsymbol{\lambda}}^{*T} \underline{\boldsymbol{A}}_h$, dove $\underline{\boldsymbol{\lambda}}^*$ è il vettore delle variabili duali ottime, $c_h = 1$ (in questo caso) ed $\underline{\boldsymbol{A}}_h \in \underline{\boldsymbol{A}} \backslash \overline{\underline{\boldsymbol{A}}}$. Il problema viene quindi formulato come:

$$\begin{aligned} \min 1 - \sum_{j=1}^{n} \lambda_j^* x_j \\ \sum_{j=1}^{n} l_j x_j \leqslant L \\ \underline{\boldsymbol{x}} \in \mathbb{Z}^n \end{aligned}$$

(a) Problema di Pricing

$$\begin{aligned} w^* = \max \sum_{j=1}^{n} \lambda_j^* x_j \\ \sum_{j=1}^{n} l_j x_j \leqslant L \\ \underline{\boldsymbol{x}} \in \mathbb{Z}^n \end{aligned}$$

(b) Problema di Pricing riscritto

Figura 13.5: Problema di Pricing

Se $w^* > 1$ allora $\underline{\boldsymbol{x}}$ è il nuovo vettore colonna da aggiungere alla matrice $\overline{\underline{\boldsymbol{A}}}$ e si ripete la ricerca dell'ottimo nel **master problem ridotto**, altrimenti se $w^* \leqslant 1$ il vettore $\underline{\boldsymbol{z}}^*$ è soluzione ottima anche del **master problem**.

La procedura di generazione di colonne però si arresta solo quando il termine $w^* \leqslant 1$, altrimenti non si ha garanzie che la soluzione identificata sia ottima anche per la formulazione **intera**.

È importante notare che il problema di **pricing** sia risolvibile non necessariamente con tecniche di programmazione lineare intera, ma essendo un tipo di **problema dello zaino** è possibile utilizzare delle tecniche, anche euristiche, ad hoc. Solo alla fine è assolutamente necessario ricorrere a metodi esatti per verificare che non esistano soluzioni accettabili.

14 Preprocessing

14.1 Qualità della formulazione

14.1.1 Programmazione lineare

Solitamente in un problema di programmazione lineare una buona metrica per la qualità della formulazione del problema è la dimensione della matrice caratteristica: una matrice più piccola è indice di un problema formulato meglio.

14.1.2 Programmazione intera

Per un problema di programmazione intera o intera mista però, la dimensione della matrice non risulta più essere una metrica valida:

1. Variabili intere e continue non possono essere considerate equamente quando si va a misurare la qualità della formulazione: variabili intere tendono ad essere molto più difficili da risolvere che con variabili continue.
2. Il grado di difficoltà aumenta esponenzialmente con il numero delle variabili.
3. Una problema intero diventa più facile da risolvere, non più difficile (come viene fatto nell'algoritmo dei piani di taglio).

In questi problemi il criterio per misurare la qualità va a basarsi sul poliedro della regione ammissibile del problema rilassato.

L'idea fondamentale del **preprocessing** è di riformulare il problema in modo tale da minimizzare la differenza dei valori tra la funzione originale intera e quella rilassata.

Definizione 14.1.1 (Poliedro). L'insieme di tutti i punti (o soluzioni) che soddisfa un set di vincoli lineari del tipo:

$$P = \{\underline{x} : \boldsymbol{A}\underline{x} \leqslant \underline{\boldsymbol{b}}, \underline{x} \text{ continuo}\}$$

È detto **poliedro**.

Definizione 14.1.2 (Formulazione). Dato un insieme $S_y = \left\{\underline{y} \in \mathbb{Z}^p : \boldsymbol{G}\underline{y} \leqslant \underline{\boldsymbol{b}}, \underline{y} \text{ intero}\right\}$, un poliedro $P \subseteq E^p$ è una **formulazione** per S_y se e solo se $S_y \subseteq P$.

Questo significa che una formulazione di un problema intero deve essere un poliedro definito sullo stesso spazio reale p-dimensionale e deve contenere lo stesso numero di punti ammissibili.

Definizione 14.1.3 (Formulazione di un problema intero). Dato $S_{xy} = \left\{(\underline{x}, \underline{y}) : \boldsymbol{A}\underline{x} + \boldsymbol{G}\underline{y} \leqslant \underline{\boldsymbol{b}}, \underline{x} \in E^n, \underline{y} \in \mathbb{Z}^p\right\}$, un poliedro $P \subseteq E^{n+p}$ sono una **formulazione** per S_{xy} se $S_{xy} \subseteq P$.

Definizione 14.1.4 (Formulazione migliore). Date due formulazioni P_1, P_2 per S_y, P_1 è una **formulazione migliore** di P_2 se $P_1 \subseteq P_2$.

Definizione 14.1.5 (Formulazione ideale). Data una formulazione $S = \left\{\underline{y} : \boldsymbol{G}\underline{y} \leqslant \underline{\boldsymbol{b}}, \underline{y} \text{ intero}\right\}$, questa è detta **ideale** se tutti i vertici del poliedro corrispondente sono **interi**.

14.2 Migliorare i vincoli in un problema intero

Supponiamo da iniziare da un vincolo del tipo $\underline{\boldsymbol{a}}^T \underline{\boldsymbol{y}} \leqslant b$. Si procede come segue:

1. Definisco $M = \sum_{a_j>0} a_j$ e $S = \{a_j : |a_j| > M - b\}$.
2. Se il set $S = \emptyset$ allora il vincolo non è migliorabile.
3. Altrimenti seleziono un termine $a_k \in S$:
   (a) Se $a_k > 0$ allora aggiorno $\overline{a}_k = M - b, \overline{b} = M - a_k$.
   (b) Altrimenti aggiorno $\overline{a}_k = b - M$.
4. Ripeto l'operazione dal punto 1.

15 SCIP

SCIP è una libreria che permette di implementare rapidamente problemi lineari interi e di risolverli con **Branch-and-cut**.

15.1 Installare SCIP

15.1.1 Installare SCIP su Arch Linux

SCIP e' gia' al interno del AUR, https://aur.archlinux.org/packages/scipoptsuite/ quindi per installarlo basta eseguire: `yaourt scipoptsuite`

15.1.2 Installare SCIP su macOs

Su sistemi **macOS** può essere installato in tre passi:

1. Installa gmp con `brew install gmp`
2. Installa gcc 7 con `brew install gcc@7`, rinominando eventualmente in gcc. Purtroppo attualmente SCIP supporta solo fortran 4 disponibile in gcc 7.
3. Scarica SCIP da http://scip.zib.de/#download per il tuo sistema operativo.
4. Estrai l'archivio e sposta i file nella tua `/usr/local`.

15.2 Installare PySCIPOpt

Si tratta di una libreria che consente di usare i metodi di SCIP comodamente da python. Dopo aver installato **SCIP** è sufficiente eseguire nel terminale per installarla `pip install pyscipopt`.

15.3 Problemi di installazione comuni

GCC non linkato Se quando esegui `brew doctor` ti suggerisce di linkare `gcc` potrebbe essere necessario eseguire i seguenti due comandi:

```
sudo chown -R $(whoami):admin /usr/local/share/man
brew link -overwrite gcc
```

16 Premessa

Nei script presentati in seguito, si assumerà presente il seguente codice:

```
import random
random.seed(42) # For reproducibility
```

```
def print_results(model, EPS=1.e-6):
    x,y = model.data
    edges = [(i,j) for (i,j) in x if model.getVal(x[i,j]) > EPS]
    facilities = [j for j in y if model.getVal(y[j]) > EPS]
    print("Optimal value=", model.getObjVal())
    print("Facilities at nodes:", facilities)
    print("Edges:", edges)
```

17

Uncapacitated facility location

$$
\begin{aligned}
z = \min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} + \sum_{i=1}^{n} f_i y_i & \\
\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in \{1, \ldots, m\} & \\
\sum_{j=1}^{m} x_{ij} \leqslant M y_j \quad \forall i \in \{1, \ldots, n\} & \\
\underline{x} \in 0,1^{n \times m} & \\
\underline{y} \in 0,1^{n} &
\end{aligned}
$$

Figura 17.1: Uncapacitated facility location

17.1 Modello

```python
from pyscipopt import Model, quicksum

def uncapacitated_facility_location(facilities_number, clients_number, fixed_costs, costs, capacity):
    model = Model("Uncapacitated facility location")
    x,y = {},{}
    # Adding the variables to the model
    for j in range(facilities_number):
        y[j] = model.addVar(vtype="B", name="y(%s)"%j)
        for i in range(clients_number):
            x[i,j] = model.addVar(vtype="B", name="x(%s,%s)"%(i,j))
    for i in range(clients_number):
        model.addCons(quicksum(x[i,j] for j in range(facilities_number)) == 1, "Demand(%s)"%i)
    for j in range(facilities_number):
        model.addCons(
            quicksum(x[i,j] for i in range(clients_number)) <= capacity[j]*y[j], "Capacity(%s)"%i
        )
    model.setObjective(
        quicksum(fixed_costs[j]*y[j] for j in range(facilities_number)) +
        quicksum(costs[i,j]*x[i,j] for i in range(clients_number) for j in range(facilities_number)),
        "minimize")
    model.data = x,y
    return model
```

17.2 Parametri

```
facilities_number = 10
clients_number = 8
min_cost = 1
max_cost = 100
default_capacity = 1
capacity = {j:default_capacity for j in range(facilities_number)}
costs = {(i,j):random.randint(min_cost, max_cost) for i in range(clients_number) for j in range(facilities_
fixed_costs = {j:random.randint(min_cost, max_cost) for j in range(facilities_number)}
```

17.3 Esecuzione

```
model = uncapacitated_facility_location(facilities_number, clients_number, fixed_costs, costs, capacity)
model.optimize()
print_results(model)
# Optimal value= 429.0
# Facilities at nodes: [0, 2, 3, 4, 5, 6, 8, 9]
# Edges: [(7, 0), (0, 2), (2, 3), (5, 4), (3, 5), (4, 6), (1, 8), (6, 9)]
```

18

Problema delle p-Mediane

Si tratta di un problema simile all'uncapacitated facility location ma cerca di posizionare p facilities ignorando i costi fissi.

$$
\begin{aligned}
z = \min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} & \\
\sum_{i=1}^{n} x_{ij} = 1 \quad & \forall j \in \{1, \ldots, m\} \\
\sum_{i=1}^{n} y_i = p & \\
x_{ij} \leqslant y_j \quad \forall i \in \{1, \ldots, n\} \quad & \forall j \in \{1, \ldots, m\} \\
\underline{x} \in 0,1^{n \times m} & \\
\underline{y} \in 0,1^{n} &
\end{aligned}
$$

Figura 18.1: Problema delle p-Mediane

18.1 Modello

```python
from pyscipopt import Model, quicksum
def p_medians(I, J, costs, p):
    model = Model("P-medians")
    x,y = {},{}
    # Adding the variables to the model
    for i in range(I):
        y[i] = model.addVar(vtype="B", name="y(%s)"%i) # Y variables are binary
        for j in range(J):
            x[i,j] = model.addVar(vtype="B", name="x(%s,%s)"%(i,j)) # X variables are binary
    for i in range(I):
        model.addCons(quicksum(x[i,j] for j in range(J)) == 1, "Demand(%s)"%i)
    for i in range(I):
        for j in range(J):
            model.addCons(x[i,j] <= y[j], "Open(%s,%s)"%(i,j))
    model.addCons(quicksum(y[j] for j in range(J)) <= p, "Median")
    model.setObjective(
        quicksum(costs[i,j]*x[i,j] for i in range(I) for j in range(J)),
        "minimize")
    model.data = x,y
    return model
```

18.2 Parametri

```
I = 10
J = 5
min_cost = 1
max_cost = 100
p = 5
costs = {(i,j):random.randint(min_cost, max_cost) for i in range(I) for j in range(J)}
```

18.3 Esecuzione

```
model = p_medians(I, J, costs, p)
model.optimize()
print_results(model)
#Optimal value= 126.0
#Facilities at nodes: [0, 1, 2, 3, 4]
#Edges: [(0, 2), (1, 4), (2, 3), (3, 2), (4, 3), (5, 0), (6, 1), (7, 0), (8, 2), (9, 2)]
```

19 Problema dei p-centri

Seleziona un dato numero di facilities da un insieme di punti in un grafo in modo tale che il valore massimo della distanza da un cliente alla facility è minima.

$$
\begin{aligned}
& z = \min l \\
& \sum_{j=1}^{m} x_{ij} = 1 \quad \forall i \in \{1,\dots,n\} \\
& \sum_{j=1}^{m} y_j = p \\
& \sum_{j=1}^{m} c_{ij} x_{ij} \leqslant l \quad \forall i \in \{1,\dots,n\} \\
& x_{ij} \leqslant y_j \quad \forall i \in \{1,\dots,n\} \quad \forall j \in \{1,\dots,m\} \\
& \underline{\boldsymbol{x}} \in 0,1^{n \times m} \\
& \underline{\boldsymbol{y}} \in 0,1^{n}
\end{aligned}
$$

Figura 19.1: Problema delle p-Mediane

19.1 Modello

```python
from pyscipopt import Model, quicksum
def p_centers(I, J, costs, p):
    model = Model("P-medians")
    x,y = {},{}
    # Adding the variables to the model
    l = model.addVar(vtype="C", name="l")
    for i in range(I):
        y[i] = model.addVar(vtype="B", name="y(%s)"%i) # Y variables are binary
        for j in range(J):
            x[i,j] = model.addVar(vtype="B", name="x(%s,%s)"%(i,j)) # X variables are binary
    for i in range(I):
        model.addCons(quicksum(x[i,j] for j in range(J)) == 1, "Demand(%s)"%i)
        model.addCons(quicksum(costs[i,j]*x[i,j] for j in range(J)) <= l, "Costs(%s)"%(i))
    for i in range(I):
        for j in range(J):
            model.addCons(x[i,j] <= y[j], "Open(%s,%s)"%(i,j))
    model.addCons(quicksum(y[j] for j in range(J)) <= p, "Center")
    model.setObjective(l, "minimize")
    model.data = x,y
    return model
```

19.2 Parametri

```
I = 10
J = 5
min_cost = 1
max_cost = 100
p = 5
costs = {(i,j):random.randint(min_cost, max_cost) for i in range(I) for j in range(J)}
```

19.3 Esecuzione

```
model = p_centers(I, J, costs, p)
model.optimize()
print_results(model)
# Optimal value= 29.0
# Facilities at nodes: [0, 1, 2, 3, 4]
# Edges: [(0, 2), (1, 4), (2, 3), (3, 2), (4, 3), (5, 0), (6, 1), (7, 0), (8, 2), (9, 2)]
```