UNIVERSITÀ DEGLI STUDI DI PISA

Facoltà di Scienze Matematiche Fisiche Naturali

Corso di Laurea in Informatica Applicata

Dipartimento di Informatica

Approcci Interior Point per il Problema di Flusso di Costo Minimo Quadratico

Tesi di Laurea Triennale

Relatore
Prof. Antonio Frangioni

Laureanda
Giulia Celi

Sessione di Laurea 24 Giugno 2013
Anno Accademico 2012-2013

A Pi e Ut

Ringraziamenti

Desidero innanzitutto ringraziare il professor Antonio Frangioni per il prezioso supporto durante il tirocinio e per avermi introdotta nel mondo della ricerca operativa, insegnandomi ad amarla ed apprezzarla. Inoltre, ringrazio mia madre Lorenza per non aver mai smesso di credere in me e mio padre Roberto per avermi spronato a puntare sempre più in alto. Ringrazio Thomas, che tra pochi mesi diventerà mio marito, per essere sempre stato al mio fianco, anche quando non era facile restarci. Ringrazio Gian-Luca e Arianna per l'affetto e il supporto dimostratomi. Ringrazio Giulia e Beatrice, le mie "sorelle acquisite", per le risate e i momenti di gioia che mi hanno regalato. Ringrazio tutti coloro che occupano un posto importante nella mia vita, rendendola quell'esperienza unica che vivo ogni giorno. Infine, con un po' di presunzione, ringrazio me stessa per la tenacia che metto in ogni cosa che faccio.

Giulia Celi.

Contents

Prefazione

Ogni giorno ci troviamo di fronte a scelte nelle quali agiamo cercando di considerare quello che riteniamo più conveniente: se, ad esempio, stiamo pianificando un viaggio, probabilmente, sceglieremo cosa mettere in valigia prestando attenzione ad avere tutto il necessario senza eccedere sul peso, cercheremo di percorrere le strade che ci porteranno a destinazione nel modo meno costoso in termini di tempo o costo ed organizzeremo le nostre giornate programmando le varie escursioni nella sequenza che riteniamo più efficiente. Ne risulta che, in maniera del tutto naturale, la nostra mente possiede insiti alcuni metri di misura con cui associa funzioni di valore e costo e cerca di trovare una soluzione che massimizzi il piacere o, specularmente, minimizzi il disagio. La Ricerca Operativa è la disciplina scientifica che si occupa proprio di questo: affrontare con gli strumenti matematici tutte le attività decisionali in cui occorre gestire e coordinare attività e risorse disponibili in quantità limitata in modo da rispettare un insieme assegnato di vincoli al fine di massimizzare o minimizzare una funzione obiettivo. Per la modellazione e la risoluzione di buona parte dei problemi decisionali si ricorre sovente all'entità matematica di grafo. Questa struttura è presente negli aspetti del mondo reale come nessun altro strumento di modellazione matematica. In molti casi non è difficile riuscire ad individuarla, si pensi banalmente alla rete stradale cui si faceva riferimento prima, in altri invece resta più oscura ed è necessaria un'adeguata preparazione per poterla riconoscere e plasmare in maniera opportuna. Ne è esempio il problema di cosa inserire nella valigia, noto in letteratura come *Knapsack Problem*, che, anche se apparentemente lontano dalla nozione di grafo, è riconducibile attraverso una serie di ipotesi ad un problema basato su grafi, lo *Shortest (/Longest) Path Problem*, che a sua volta può essere formulato come uno tra i più generali problemi di ottimizzazione, il *Minimum Cost Flow Problem*

Introduzione

Il classico problema di flusso di costo minimo con funzione costo lineare si estende facilmente, dal punto di vista algebrico, al caso in cui il costo sia quadratico convesso separabile sugli archi. Alcuni algoritmi di ottimizzazione riescono a trattare con modifiche minori tale estensione; è questo il caso dei metodi del punto Interno. Questi metodi costituiscono la principale alternativa ai metodi del simplesso per la risoluzione di problemi di programmazione lineare. Essi hanno complessità polinomiale e possono essere generalizzati a problemi di programmazione non lineare convessa. Il presente lavoro di tesi si focalizza sullo studio di una versione specializzata del metodo del Punto Interno al problema di flusso di costo minimo quadratico.

Chapter 1

Metodi Interior Point

1.1 Cenni storici

I metodi del punto interno costituiscono il nucleo di molti solutori popolari per l'ottimizzazione lineare e non lineare. Nell'ambito della programmazione lineare, tuttavia, questo non sempre accade a causa del dominio assoluto del metodo del simplesso. Quest'ultimo venne inventato da Dantzig nel 1947 e consiste in un metodo iterativo durante il quale, ad ogni passo, ci si muove da un vertice all'altro sino a trovare quello ottimo, o sino a sancire l'illimitatezza del problema. Nonostante nella pratica funzioni piuttosto bene, è chiaro che potenzialmente quest'algoritmo potrebbe visitare tutti i vertici del poliedro. Il fatto è che, pur essendo nella maggior parte dei casi polinomiale, al caso pessimo la complessità del simplesso è esponenziale nella dimensione del problema. È cresciuto così l'interesse per la ricerca di un metodo dalla complessità polinomiale. Nel 1984 Karmarkar presentò un nuovo algoritmo polinomiale [5] che, in opposizione alla metodologia precedente, non si muove piè sui vertici ma all'interno della regione ammissibile. Ha così inizio alla rivoluzione del punto interno, che come in tutte le rivoluzioni, include le vecchie idee riviste o ricosperte sotto una nuova luce.

1.2 Idea di base

Si consideri un problema con soli vincoli di disuguaglianza

$$\min \left\{ f(x) \ : \ g_i(x) \geq 0 \quad i = 1, ..., m \right\}.$$

Al fine di trovare una buona strategia risolutiva è bene chiedersi in prima istanza cosa rende questo problema difficile rispetto al caso non vincolato. Banalmente viene spontaneo rispondere i "vincoli". Forti della nostra deduzione, li rimuoviamo, prestando attenzione ad inserirli opportunamente in funzione obiettivo

$$f(x) + F(g(x), \mu)$$

dove μ può essere interpretato come un vettore dei pesi, ed F come una funzione di merito che misura, in un certo senso, l'inammissibilità della soluzione corrente.

L'idea consiste nel ricondurre la soluzione di un problema vincolato a quella di un problema non vincolato. Lo schema algoritmico che scaturisce è di tipo sequenziale, ovvero basato sulla soluzione di una successione di problemi non vincolati, in modo tale che le soluzioni convergano a quella di un problema di vincolato.

Procedure MetodoIterativo (x_0, μ_0, F, g)
begin
k=0
while (CriterioArresto) **do**{
begin
$x_{k+1} = \arg\min_{x\in\mathcal{F}}$ f(x)+ F(g(x),μ_k)
μ_{k+1}=AggiornaPeso(μ_k)
k++}
end
end.

Si osservi che la procedura appena esposta ha senso se esiste un metodo tale per cui il problema modificato sia facile da risolvere e sia possibile trovare una regola di aggiornamento del peso (`AggiornaPeso`), un valore del vettore dei pesi, e una funzione F tali che il metodo converga. Sulla base di come vengono fatte queste scelte è possibile distinguere due grandi famiglie di metodi

- *Metodi a Penalità*, in cui la violazione dei vincoli imputati penalizza la funzione obiettivo
- *Metodi a Barriera*, in cui si penalizza l'avvicinarsi ai vincoli imputati all'interno della regione ammissibile

Nei metodi a penalità si fa uso di una funzione continua $\rho(x)$, detta di penalità, tale che $\rho(x)$ è nulla nei punti che rispettano i vincoli e maggiore di zero altrove. Nei metodi a barriera la funzione di merito che viene impiegata è la cosiddetta funzione barriera. Questa funzione gode delle seguenti proprietà:

- definita e positiva sull'insieme ammissibile $\mathcal{F}$
- $x_i \in \mathcal{F},\ \lim_{i\to\infty} x_i\ = x \in \sigma\mathcal{F}\ \implies\ \lim_{i\to\infty} B\left(x_i\right) = +\infty$

il sottoproblema risultante è

$$\min \beta(x) = f(x) + \mu B(g(x))$$

dove μ è detto parametro barriera. Si tratta di un modello non vincolato dove si è creato un effetto barriera che impedisce ad un punto interno, ovvero un punto $x_{int}\ \in \mathcal{F}_{int} = \{x \in R^n : g\left(x\right) < 0\}$, di uscire dalla regione ammissibile. Si noti

che il valore di μ influenza l'effetto della barriera sulla funzione f, in particolare al crescere di questo valore cresce anche l'effetto barriera. Contrariamente al metodo delle funzioni di penalità, si presti attenzione a fatto che il metodo barriera lavora sempre con punti ammissibili, ovvero punti che si trovano in $\mathcal{F}_{int}$ ed è per questo motivo che questo metodo sia applicabile solo nell'ipotesi che quest'insieme non sia vuoto o, in altri termini, solo a problemi la cui regione ammissibile non sia descritta da soli vincoli di uguaglianza. Inoltre, il fatto che si lavori sempre su punti ammissibili implica che anche il punto di partenza lo sia e questo non è sempre facile da ottenere.

Fra le funzioni di barriera più comuni troviamo la barriera inversa, ovvero una funzione continua che tende ad infinito al tendere 0 di $g(x)$ in ciascun vincolo

$$B(x) = -\sum_{i=1}^{m} \frac{1}{g(x)}$$

e la barriera logaritmica

$$B(x) = -\sum_{i=1}^{m} \ln(g(x))$$

più diffusa e storicamente più importante. Il metodo barriera genera una sequenza di punti

$$x_k = \arg\min_{x \in \mathcal{F}} f(x) + \mu_k B(x), \quad k = 0,\ 1, \ldots$$

La curva descritta da questi punti prende il nome di percorso centrale, o central path. L'algoritmo risultante è

```
Procedure MetodoBarriera (x0, μ0 )
  begin
   k=0
   while (CriterioArresto) do
     begin
       x_{k+1} = arg min_{x∈F} f(x)+ μ_k B(x)
      μ_{k+1}=AggiornaPeso(μ_k)
      k++
     end
  end.
```

1.3 Il metodo di Newton

Il metodo di Newton è una procedura iterativa utilizzata per risolvere un'equazione non lineare

$$F(t) = 0.$$

Ad ogni iterazione di Newton viene calcolata la nuova iterata, t_{k+1}, a partire dalla precedente, t_k, secondo la formula

$$t_{k+1} = t_k - \nabla f(t_k)^{-1} F(t_k).$$

Assumendo che il sistema abbia soluzione t^*, è possibile approssimare la funzione con un polinomio utilizzando lo sviluppo di Taylor su t_k

$$F(t) = F(t_k) + \nabla f(t_k)(t - t_k) + (t - t_k)^T \nabla^2 f(t_k) \frac{t - t_k}{2} + \dots$$

osservando che t_{k+1} può essere alternativamente visto come la radice radice dei due termini dello sviluppo di Taylor, possiamo scrivere

$$0 = M_k(t_{k+1}) = F(t_k) + \nabla f(t_k)(t - t_k).$$

Sia $\triangle t_k = t_{k+1} - t_k$, possiamo riscrivere l'iterata di Newton come

$$\nabla f(t_k) \triangle t_k = -F(t_k). \tag{1.1}$$

Il metodo di Newton si estende in modo ovvio ai sistemi di equazioni non lineari, per maggiori dettagli consultare [1].

1.4 Il metodo Interior Point

Un metodo del punto interno (*IPM*, in breve) è un potente strumento per risolvere problemi di programmazione lineare, non lineare. A seconda di come viene effettuato il passo di Newton, è possibile distinguere tra metodo primale, duale e primale duale. In questa tesi, ci concetreremo soltanto sui metodi primali-duali, descrivendo superficialmente le versioni primale e duale "pure".

Un metodo primale-duale si applica alla formulazione primale-duale della programmazione lineare.

$$\min \{\ cx\ :\ Ax = b\ ,\ x \geq 0\ \} \tag{1.2}$$

$$\max \{\ by\ :\ Ay + s = c\ ,\ s \geq 0\ \} \tag{1.3}$$

dove $A \in \mathbb{R}^{m \times n}$, $x, s, c \in\ \mathbb{R}^n$ e $y, b \in \mathbb{R}^m$. Il motivo per il quale si è deciso di approfondire sui metodi primali-duali è che questi, rispetto agli altri approcci, si sono mostrati più veloci ed affidabili. L'usuale trasformazione nei metodi del punto interno consiste nell'eliminare i vincoli di disuguaglianza, inserendoli in funzione obiettivo per mezzo della barriera logaritmica. Il problema primale risulta quindi

$$\begin{aligned} \min cx\ & - \mu \sum_{i=1}^{k} \ln(x_i) \\ & Ax = b \end{aligned}$$

dove $\mu > 0$ è il paramentro barriera. La Lagrangiana, vedi definizione al paragrafo B.2, associata al problema è

$$L(x, \mu, y) = cx - \mu \sum_{i=1}^{k} \ln(x_i) - y(Ax - b)$$

per risolvere il problema primale, è necessario trovare la soluzione del sistema KKT[1]

$$Ax = b$$
$$A^T y + s = c$$
$$XSe = 0$$
$$(x, s) \geq 0$$

dove $X = diag\,(x)\,,\ S = diag\,(s)$ ed $e \in \mathbb{R}^n$ è un vettore di uni[2]. L'approccio IP consiste nel cercare la soluzione ottima muovendosi all'interno della regione ammissibile seguendo la traiettoria dei punti individuata dal central path, ovvero seguendo la curva

$$C = \{(x, y, s) \in \mathcal{F}^0 : x_i s_i = \tau,\ \forall i = 1, \ldots, n\}$$

dove $\mathcal{F}$ è l'insieme ammissibile della coppia primale-duale, che assumiamo non vuoto (sennò il problema non avrebbe soluzione)

$$\mathcal{F} = \{(x, y, s) : Ax = b,\ Ay + s = c,\ (x, s) \geq 0\}$$

individuata dall'insieme di soluzioni del sistema di equazioni

$$\begin{array}{c} Ax - b = 0 \\ Ay + s - c = 0 \\ XSe = \tau e \\ (x, s) \geq 0 \end{array} \quad . \tag{1.4}$$

Queste condizioni differiscono dalle KKT solo nel termine τ, in particolare, notiamo l'equazione $XSe = \tau e$ può essere interpretata come una perturbazione della condizione di complementarità del problema originale. Si osservi che il percorso centrale è ben definito in quanto il sistema (1.4) ammette unica soluzione per ogni $\tau > 0$. Inoltre, i punti della traiettoria convergono ad una soluzione primale-duale quando $\tau \to 0$. Sia dunque τ una quantità inferiore o al più uguale al parametro penalità della funzione barriera, μ

$$\tau = \sigma\mu$$

[1] vedi teorema 31

[2] Ricordiamo che con $diag(a_1 \ldots a_k)$ si intende la matrice diagonale i cui valori sulla diagonale principale sono $a_1 \ldots a_k$ e, analogamente,con $diag(a)$, dove $a \in \mathbb{R}^n$, si indica la matrice diagonale i cui valori sulla diagonale principale sono gli elementi del vettore a

dove $\sigma \in [0,1]$ è detto parametro di centraggio e, come vedremo, giocherà un ruolo fondamentale nell'efficacia dell'algoritmo. Non soddisfare subito la complementarità di problema di partenza può essere visto come posporre il lato combinatorio, e quindi difficile, di un probema di ottimizzazione vincolata. Il modello algoritmico risultante quindi, non solo previene, per mezzo della funzione barriera, l'attivazione dei vincoli di disuguaglianza, ma risolve iterativamente il problema modificato per diversi valori di complementarità ovvero risolve ad ogni iterazione il sistema KKT del problema modificato per diversi valori τ, e quindi di μ. Riscriviamo il precedente sistema come segue

$$F(t) = \begin{bmatrix} Ax-b \\ Ay+s-c \\ XSe-\sigma\mu e \end{bmatrix} = 0, \quad x > 0, \quad s > 0$$

dove $t = (x, y, s)$. La maggior parte degli algoritmi IP, tra cui il nostro, adottano il metodo di Newton per spostarsi da un punto all'altro del percorso centrale, calcolando ad ogni passo la direzione secondo la formula

$$\nabla f(t) \triangle t = -F(t)$$

dove $\nabla F(t)$ è la derivata di $F(t)$. Quanto detto equivale a risolvere alla k-esima iterazione

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S_k & 0 & X_k \end{pmatrix} \begin{pmatrix} \triangle_{x_k} \\ \triangle_{y_k} \\ \triangle_{s_k} \end{pmatrix} = - \begin{pmatrix} Ax_k - b \\ A^T y_k + s_k - c \\ X_k S_k e - \sigma_k u_k e \end{pmatrix} = \begin{pmatrix} \varepsilon_p \\ \varepsilon_d \\ \varepsilon_u \end{pmatrix}$$

partendo da una tripla iniziale $(x_0, y_0,\ s_0) > 0$. Si pone quindi

$$(x_{k+1}, y_{k+1}, s_{k+1}) \leftarrow (x_k, y_k, s_k) + (\alpha_k^P \triangle x_k, \alpha_k^D \triangle y_k,\ \alpha_k^D \triangle s_k)$$

dove α_k^P e α_k^D indicano la dimensione dei passi nello spazio del primale ed in quello del duale, rispettivamente. Il sistema appena visto può essere ricondotto ad una forma lineare a bloccchi detta sistema aumentato di dimensione $(n+m)$. Infatti, essendo X una matrice diagonale invertibile, è possibile, usando la terza equazione, eliminare il vettore

$$\triangle_s = X^{-1}(\varepsilon_u - S\triangle_x) = X^{-1}\varepsilon_u - X^{-1}S\triangle_x$$

dalla seconda equazione. Introducendo quindi la matrice diagonale

$$\theta = XS^{-1} = S^{-1}X \qquad \Longrightarrow \qquad \theta^{-1} = X^{-1}S$$

e sostituendola nell'equazione precedente si ottiene

$$\triangle_s = X^{-1}\varepsilon_u - X^{-1}S\triangle_x = X^{-1}\varepsilon_u - \theta^{-1}\triangle_x = -\theta^{-1}\triangle_x + X^{-1}\varepsilon_u$$

che sostituito nella seconda equazione

$$A^T\triangle_y + \triangle_s = \varepsilon_d \qquad \Longleftrightarrow \qquad A^T\triangle_y - \theta^{-1}\triangle_x + X^{-1}\varepsilon_u = \varepsilon_d \qquad \Longleftrightarrow$$

$$-\theta^{-1}\triangle_x + A^T\triangle_y = \quad \varepsilon_d - X^{-1}\varepsilon_u$$

unendo il risultato appena ottenuto con la prima equazione è possibile ottenere il seguente sistema aumentato simmetrico

$$\begin{pmatrix} -\theta^{-1} & A^T \\ A & 0 \end{pmatrix}\begin{pmatrix} \triangle_x \\ \triangle_y \end{pmatrix} = \begin{pmatrix} \varepsilon_d - X^{-1}\varepsilon_u \\ \varepsilon_p \end{pmatrix} = \begin{pmatrix} r \\ h \end{pmatrix}$$

dove, per semplicità, è stato omesso l'indice dell'iterazione. Compiendo ora i seguenti passi

$$\left\{\begin{array}{c} A\ \triangle_x = h \\ -\theta^{-1}\triangle_x + A^T\triangle_y = r \end{array}\right. \iff \left\{\begin{array}{c} \triangle_x\ = A^{-1}h \\ -\theta^{-1}\triangle_x = -A^T\triangle_y + r \end{array}\right. \iff$$

$$\iff \left\{\begin{array}{c} \triangle_x\ = A^{-1}h \\ \triangle_x = \theta A^T\triangle_y - \theta r \end{array}\right. \iff A^{-1}h = \theta A^T\triangle_y - \theta r \iff$$

$$\iff \boxed{(A\theta A^T)\triangle_y = A\theta r + h = g}$$

si ottiene il sistema di equazioni normali di dimensione m, simmetrico e definito positivo.

1.5 Estensione al caso quadratico

Il metodo interior point primale-duale si estende facilmente a problemi di programmazione convessa-quadratica (QP) [6] [7]. Enunciamo un problema di QP come

$$\min\left\{\ cx\ + \frac{1}{2}\,x^TQx:\ Ax = b\ ,\ x \geq 0\ \right\}$$

dove $Q \in \mathbb{R}^{n\times n}$ è una matrice semidefinita positiva, $A \in \mathbb{R}^{m\times n}$ è la matrice dei vincoli lineari e i vettori $x, c \in \mathbb{R}^n$ e $b \in \mathbb{R}^m$. Come nel caso lineare, rimpiazziamo i vincoli di disuguaglianza con la barriera logaritmica

$$\min\ cx\ + \frac{1}{2}\,xQx\ - \mu\sum_{i=1}^{n} ln(x_i)$$

e calcoliamone la funzione lagrangiana

$$L(x,\mu,y) = cx\ + \frac{1}{2}\,xQx\ - y(Ax-b)\ - \mu\sum_{i=1}^{n} ln(x_i)$$

con $\mu > 0$. Ricaviamo ora le condizioni di Karush-Kuhn-Tucker

$$\begin{array}{c} Ax = b \\ Ay + s - Qx = c \\ XSe = \mu e \\ (x,s) \geq 0. \end{array} \tag{1.5}$$

Applichiamo ora il metodo di Newton al sistema 1.5, si ottiene

$$\begin{pmatrix} A & 0 & 0 \\ -Q & A^T & I \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} \xi p \\ \xi d \\ \xi u \end{pmatrix} \tag{1.6}$$

dove $\xi p = b - Ax$, $\xi d = c - A^T y - s + Qx$, $\xi u = \mu e - XSe$. Come nel caso lineare, usando la terza equazione, si elimina dalla seconda il vettore

$$\Delta s = X^-1(\xi u - S\Delta x) = -X^{-1}S\Delta x + X^{-1}\xi u$$

ottenendo il seguente sistema aumentato $(n + m)$

$$\begin{pmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} \xi d - X^{-1}\xi u \\ \xi p \end{pmatrix} \tag{1.7}$$

dove $\Theta = XS^{-1}$. Eliminando $\triangle_x$ dalla prima equazione possiamo ridurre (1.7) alla seguente forma

$$(A(Q + \theta^{-1})^{-1}A^T)\triangle_y = b_{QP}. \tag{1.8}$$

La procedura risolutiva del caso quadratico è identica a quella per il caso lineare.

1.6 Risoluzione del sistema KKT

Abbiamo visto, quindi, che ogni passo di un metodo del punto interno richiede la soluzione di un sistema di equazioni lineari potenzialmente grande e, quasi sempre, sparso del tipo

$$\begin{pmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle y \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{1.9}$$

L'uso della barriera logaritmica provoca un malcondizionamento del sistema aumentato (1.9): mano a mano che ci si avvicina all'ottimo, infatti, mentre alcuni elementi di Θ tenderanno a zero, altri tenderanno ad infinito. Se il sistema viene risolto con i metodi iterativi, il malcondizionamento potrebbe essere un inconveniente importante [3] e, per affrontarlo, si trasforma (1.9) in un sistema equivalente con proprietà spettrali più favorevoli [4]. Per farlo, s'introduce una matrice di precondizionamento M, altresì detta più semplicemente precondizionatore.

$$M^{-1} \begin{pmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle y \end{pmatrix} = M^{-1} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Esistono diverse classi di precondizionatori, più o meno efficaci a seconda della tipologia di problema in esame. Ad esempio, in fluidodinamica sono popolari i precondizionatori triangolari o diagonali a blocchi, mentre in ottimizzazione

[3] Non lo è ad esempio per la fattorizzazione di Cholesky e per i metodi diretti

[4] In realtà non è detto che si precondizioni il sistema esteso, per esempio i nostri precondizionatori lavorano sul sistema 1.8

sono più famosi i cosiddetti CP, Costraint Preconditioners. Quanto detto evidenzia il fatto che, sfruttando le caratteristiche peculiari del problema che ci si appresta a risolvere, è possibile studiarne precondizionatori "su misura". Nel corso della nostra trattazione illustreremo, esclusivamente, i precondizionatori per il problema di flusso di costo minimo, per uno studio più generale e più approfondito si rimanda a letture specifiche [2], [4].

Chapter 2

Metodi IP per il problema di flusso di costo minimo

2.1 Il problema di flusso di costo minimo

Il problema di distribuzione di flusso a costo minimo è un problema di grande rilevanza in ambito economico. Si consideri un'impresa di produzione e distribuzione di un certo bene composta da diversi centri:

- centri di *produzione* in cui la merce, non solo transita, ma viene anche realizzata in quantità prefissata
- centri di *distribuzione* in cui il prodotto non si limita a transitare, ma viene anche consegnato in quantità prestabilita
- centri di *stoccaggio* in cui il bene si limita a transitare

Come è ragionevole pensare, gli impianti sono connessi fra loro attraverso una serie di collegamenti, ai quali è facile associare un costo di trasporto e, in determinati contesti, si faccia riferimento ad esempio ad una rete di distribuzione elettrica, ulteriori informazioni circa la quantità massima e minima di prodotto trasportabile lungo essi. Il problema che si va ad affrontare nasce dall'esigenza di determinare il modo più conveniente con il quale è possibile traportare le merci, dagli impianti di produzione a quelli di distribuzione, all'interno della rete di collegamento appena descritta.

Come anticipato al capitolo precedente, questo problema racchiude al suo interno una quantità enorme di casistiche. In prima analisi, pur avendo qui discusso il caso di una rete di collegamento tra i centri di un'azienda, un problema di questo tipo può insorgere in altri tipi di rete, come una rete di comunicazione o una rete idraulica. Non solo, una modellazione di questo tipo è applicabile anche a situazioni che a prima vista potrebbero non apparire immediatamente riconducibili a problemi di flusso. è questo il caso di una compagnia aerea che

deve stabilire i turni del personale di terra in base al flusso di passeggeri durante le ore della giornata, rispettando le normative contrattuali che prevedono vincoli sulla durata delle turnazioni, sul numero di unità di personale ogni quantitativo prestabilito di passeggeri e sulla retribuzione dei turni. Seppur evidentemente lontano dalla nozione di rete, è possibile ricondurre questo problema ad uno di flusso di costo minimo. Non trattandosi questa di una tesi sulla modellazione matematica, si rimanda ad opportune letture per l'approfondimento della trasformazione, [3].

Per slegarci dai dettagli non significativi delle applicazioni in cui sorgono questi problemi, viene impiegata l'entità di rete. Con questo termine si indica un grafo $G = (N, A)$ pesato, cioè ai cui nodi e/o archi sono associati valori numerici detti pesi. In questo contesto, i pesi degli archi rappresentano capacità e costi, mentre quelli dei nodi la quantità dei beni che entrano nella rete, o ne escono, in quei nodi. Più precisamente: ad ogni nodo $i \in N$ è associato un valore reale b_i, che può essere:

- positivo, e in tal caso, rappresenta la quantità di bene che esce dalla rete al nodo i; b_i è allora detto domanda del nodo, ed il nodo viene detto destinazione, pozzo o nodo di output;
- negativo, e in tal caso, rappresenta la quantità di bene che entra nella rete al nodo i; $-b_i$ è allora detto offerta del nodo, ed il nodo viene detto origine, sorgente o nodo di input;
- nullo, ed in questo caso i viene detto nodo di trasferimento;

ad ogni arco $a_k = (i, j)$ sono associati un costo c_k (o c_{ij}), che indica il costo che viene pagato per ogni unità di bene che attraversa l'arco, ed una capacità inferiore l_k (o l_{ij}) e superiore u_k (o u_{ij}), che indicano, rispettivamente, il minimo ed il massimo numero di unità di bene che possono attraversare l'arco. In molte applicazioni, tra cui la nostra, la capacità inferiore viene assunta uguale a 0, e quindi viene fornita tra i parametri della rete solo la capacità superiore. Nei problemi di flusso la domanda globale, cioè la somma di tutte le domande, è uguale all'offerta globale, cioè alla somma, in valore assoluto, di tutte le offerte. In altre parole, il vettore b, detto vettore dei bilanci dei nodi, deve soddisfare la relazione :

$$\sum_{i \in N} b_i = 0 \tag{2.1}$$

Data una rete $G = (N, A)$, con $|A| = m$, un vettore $x \in \mathbb{R}^n$ è un flusso su G se verifica i seguenti vincoli di conservazione del flusso:

$$\sum_{(j,i) \in BS(i)} x_{ji} \; - \sum_{(i,j) \in FS(i)} x_{ij} \; = \; b_i \qquad i \in N \tag{2.2}$$

dove $BS(i)$ ed $FS(i)$ sono, rispettivamente, la stella entrante e la stella uscente del nodo i. In forma vettoriale i vincoli di conservazione del flusso sono

$$Ex = b \tag{2.3}$$

dove E è la matrice di incidenza[1] del grafo e $b = (b_i)$. Il valore x_k (o x_{ij}) è detto flusso dell'arco $a_k = (i, j)$. Un flusso è detto ammissibile se sono verificati i seguenti vincoli di capacità sugli archi:

$$l_{ij} \leq x_{ij} \leq u_{ij} \quad (i,j) \in A \tag{2.5}$$

Tali vincoli possono essere scritti in forma vettoriale come segue

$$l \leq x \leq u \tag{2.6}$$

dove $l = (l_{ij})$ e $u = (u_{ij})$. Dato un flusso x, il costo del flusso è dato dalla somma dei flussi degli archi per il loro costo:

$$cx = \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{2.7}$$

Si arriva così alla formulazione del problema del flusso di costo minimo con funzione di costo lineare (*MCF*, da *Min Cost Flow* problem):

$$\begin{aligned} &\min \sum_{(i,j) \in A} c_{ij} x_{ij} \\ \sum_{(j,i) \in BS(i)} x_{ji} \; - \sum_{(i,j) \in FS(i)} x_{ij} \; &= \; b_i \qquad i \in N \\ l_{ij} \leq x_{ij} &\leq u_{ij} \quad (i,j) \in A \end{aligned} \tag{2.8}$$

che, in forma matriciale, può essere alternativamente espresso come

$$\min \{ \; cx \; : \; Ex = b \; , \; l_{ij} \leq x \leq u \; \} \; , \tag{2.9}$$

dove E è la matrice di incidenza nodo-arco della rete. Nel corso della trattazione assumeremo, senza perdità di generalità, $l_{ij} = 0$, $\forall (i,j) \in A$; ovvero tratteremo il problema

$$\min \{ \; cx \; : \; Ex = b \; , \; 0 \leq x \leq u \; \} \; . \tag{2.10}$$

2.2 Il caso quadratico

Il problema appena visto (2.9) si estende facilmente al caso quadratico, in cui gli archi hanno, appunto, costi quadratici e la quantità di flusso su un arco non influenza il costo del flusso su un diverso arco. Il problema appena definito viene formulato come

[1] ricordiamo che, dato un grafo orientato $G = (N, A)$ la sua matrice d'incidenza $E = [e_{ik}]$ è una matrice $n \times m$ (le righe corrispondono ai nodi, le colonne agli archi) così definita

$$e_{ik} = \begin{cases} -1 & \text{se } i \text{ è la coda dell'arco } a_k \\ 1 & \text{se } i \text{ è la testa dell'arco } a_k \\ 0, & \text{altrimenti} \end{cases} \tag{2.4}$$

$$\begin{aligned} \min \sum_{(i,j)\in A} c_{ij}x_{ij} \; + \; \frac{q_{ij}}{2}(x_{ij})^2 \\ \sum_{(j,i)\in BS(i)} x_{ji} \; - \sum_{(i,j)\in FS(i)} x_{ij} \; = \; b_i \quad i \in N \\ l_{ij} \le x_{ij} \le u_{ij} \quad (i,j) \in A. \end{aligned} \tag{2.11}$$

Per ogni arco la funzione costo $f_{ij}(x_{ij})$ è quadratica convessa: $q_{ij} \geq 0$. Si indichino con A_Q ed A_L i sottoinsiemi disgiunti dell'insieme degli archi A contenenti, rispettivamente, gli archi con componente di costo quadratica non nulla ($q_{ij} > 0$) e quelli di costo unicamente lineare ($q_{ij} = 0$), osservi che se $A_Q = \emptyset$ allora (2.11) modella la versione lineare, viceversa, se $A_Q = A$, modella il problema quadratico "puro". Come per il modello lineare, anche il quadratico può essere ricondotto ad una formulazione compatta, del tipo

$$\min \{ \; cx + \frac{1}{2}x^T Qx \; : \; Ex = b \; , \; 0 \le x \le u \; \} \; , \tag{2.12}$$

dove abbiamo assunto $l_{ij} = 0 \;\; \forall (i,j) \in A$.

2.3 Algoritmi del punto interno

Riscriviamo il problema di flusso di costo minimo (2.10), eliminando i soli vincoli di disuguaglianza $x \leq u$ tramite variabili di slack

$$\min \{ \; cx \; : \; Ex = b \; , \; x + s = u \; , \; x, s \geq 0 \; \} \tag{2.13}$$

e ricaviamone il duale

$$\max \{ \; yb - wu \; : \; yE + z - w = c \; , \; z, w \geq 0 \; \} \; , \tag{2.14}$$

dove y, z e w sono, rispettivamente, le variabili duali dei vincoli $Ex = b$, le variabili di scarto del duale e le variabili duali dei vincoli $x \leq u$. Le condizioni KKT della suddetta coppia di problemi duali sono

$$\begin{gathered} Ex = b \\ x + s = u \\ yE + z - w = c \\ XZe = \mu e \\ SWe = \mu e \\ (x, s, z, w) \geq 0 \end{gathered} \tag{2.15}$$

dove μ è parametro, e è un vettore di uni, e le lettere maiuscole corrispono ad una matrice diagonale avente sulla diagonale gli elementi del corrispondente vettore dalla lettera minuscola. Compiamo i soliti passi anche per la versione quadratica. Consideriamo il problema (2.12) e il suo duale

$$\max \{ \; b^T y - \frac{1}{2}x^T Qx - w^T u \; : \; A^T y - Qx + z - w = c \; , \; z, w \geq 0 \; \} \; , \tag{2.16}$$

ricaviamone quindi le condizioni KKT

$$\begin{gathered} Ex = b \\ x + s = u \\ yE - Qx + z - w = c \\ XZe = \mu e \\ SWe = \mu e \\ (x, s, z, w) \geq 0. \end{gathered} \tag{2.17}$$

Come mostrato al capitolo precedente, gli algoritmi barriera cercano di trovare le soluzione ottime seguendo il percorso centrale, ovvero eseguendo una versione smorzata del metodo di Newton al sistema (2.15) nel caso lineare, o al sistema (2.17) nel caso quadratico. Tali algoritmi possono essere classificati in primali, duali o primali/duali a seconda di come viene effettuato il passo di Newton.

2.4 Risoluzione del sistema KKT

Come preannunciato nell'esposizione dei metodi IP, il calcolo più oneroso è dato dalla risoluzione del cosiddetto sistema "core", il sitema KKT (1.9). Poichè la forma del suddetto è indipendente dalla variante di algoritmo IP impiegato (primale, duale o primale-duale), il solutore del sistema KKT viene utilizzato indifferentemente in tutte le versioni disponibili.

Il sistema può essere risolto attraverso la fattorizzazione di Cholesky[2], un metodo computazionalmente efficace e numericamente stabile che, tuttavia, possiede un grosso inconveniente conosciuto sotto il nome di fenomeno del fill-in (riempimento). Infatti, quando il numero degli elementi non nulli di $K = A\Theta A^T$ è molto inferiore a quello degli elementi nulli, vengono generati elementi non nulli in corrispondenza di quelli nulli della matrice di partenza. Per minimizzare questo fatto gli algoritmi IP solitamente sfruttano, solo all'inzio della computazione, euristiche per il riordinamento di K; tuttavia, salvo alcuni casi specifici, il fenomeno non può essere del tutto evitato e, per questo motivo, sono stati proposti metodi di risoluzione alternativi.

Il più gettonato è il metodo del gradiente coniugato precondizionato (PCG), dove la scelta fondamentale è quella della matrice di condizionamento che s'intende utilizzare. L'idea alla base di qualsiasi precondizionatore, per il problema che stiamo affrontando, consiste nell'estrarre un opportuno sottografo $S = (N, A_S)$ di G tale che la matrice di partenza "ridotta" a G, vale a dire,

$$M_S = E_S \Theta_S E_S^T \ , \tag{2.18}$$

dove E_S e Θ_S denotano rispettivamente le matrici E e Θ limitate alle colonne relative agli archi in S, sia, nel contempo, facile da invertire e contenga la maggior parte delle informazioni del sistema di partenza. Dal momento che

[2]La fattorizzazione di Cholesky può essere considerata come un caso speciale della decomposizione LU e consiste nella fattorizzazione di una matrice hermitiana e definita positiva in una matrice triangolare inferiore e nella sua trasposta coniugata

l'idea appena esposta lascia potenzialmente fuori una parte delle informazioni di K, si estende (2.18) a

$$M'_S = M_S + r\ diag(M - M_S) \tag{2.19}$$

dove $diag(A)$ è la matrice diagonale avente sulla diagonale gli elementi di A, r è un parametro che può essere scelto in base alla struttura dell'istanza da risolvere. Chiaramente, (2.19), pur incorporando le informazioni su tutti gli archi, non è più costosa da invertire rispetto (2.18) e, per questo motivo, talvolta risulta più efficace su alcune classi di istanze. Giacchè (2.18) deve essere invertita a basso costo, è necessario che sia composta da elementi perlopiù nulli ovvero che S sia un grafo triangolato [3]. Poichè gli alberi sono indubbiamente grafi di questo tipo, scaturisce la classe dei cosiddetti precondizionatori basati su alberi che scelgono S come un albero di copertura di costo minimo approssimato (MST, Minimum Spannig Tree) di G, dove il peso di ogni arco (i, j) è dato dal corrispondente valore di $\theta_{i,j}$. Da quest'intuizione è possibile derivare una classe più generale di precondizionatori basata sui sottografi, che estende la famiglia precedente. L'idea è quella di estrarre da G un sottografo che contenga, possibilmente in modo stretto, un albero di copertura. Si definisce allo scopo una nuova classe di grafi, chiamata "Brother-Connected Trees" (BCT) [8], con la proprietà che il precondizionatore associato può essere fattorizzato senza fill-in.

Definizione 1 (Brother-connected tree) *Un sottografo $S = (N, A_S)$ di G è un brother-connected tree (BCT) se è un albero di copertura $T = (N, A_T)$ di G o contiene un albero di copertura T di G tale che il sottografo $S' = (N, A_S\ A_T)$ ottenuto rimuovendo tutti gli archi di T da S è formato da un certo numero $k \geq 1$ di componenti connesse[4] di nodi disgiunti $S'_1 = (N_1, A_1), ..., S'_K = (N_K, A_K)$ tali che tutti i nodi in N_i sono "fratelli" [5] in T, ed ogni S'_i è un BCT.*

Il software da noi presentato offre, oltre la classica fattorizzazione di Cholesky, il metodo PCG proponendo sia i precondizionatori ad albero che i BCT, nelle forme "standard" (2.18) e in quella con il termine diagonale (2.19). Per quanto concerne il calcolo dell'albero di copertura è possibile scegliere tra l'algoritmo di Kruskal e quello di Prim.

2.5 Formule di crash start

Come ogni approccio iterativo, anche gli algoritmi del punto interno necessitano di una soluzione di partenza per avviarsi; pertanto, a seconda del tipo di algoritmo che si intende usare, sarà necessario fornire una soluzione primale x, duale (y, z, w) o primale e duale. Nella nostra applicazione è possibile impostare diverse tipologie di formule di crash start.

[3] un grafo non orientato si dice triangolato se ogni ciclo di lunghezza maggiore o uguale a 4 possiede una corda.

[4] una componente connessa di un grafo G è un sottografo G' tale per cui esiste un cammino semplice per ogni coppia di vertici di G'

[5] figli dello stesso nodo

2.5.1 Inizializzazione variabili duali

Per le variabili duali y sono state proposte le seguenti quattro regole:

Y0) $y^0 = 0$

Y1) $y^0 = (ESE^T)^{-1}Ec$

Y2) $y^0 = b\|c\|_\infty / \max(\|b\|_\infty, 1)$

Y3) $y^0 := (ESE^T)^{-1}(b + ES(c - \tau_y u))$

dove τ_y è un parametro fissato ed $S = I$ nel caso lineare o $S = (Q + 2\tau_y I)^{-1}$ nel caso quadratico. Le regole **Y0** e **Y2** sono semplici inizializzazioni "quick&dirty" [6].La regola **Y1** equivale a scegliere la soluzione y che "viola il meno possibile" il vincolo $yE = c$ (da sola, ossia assumento z = w = 0), ovvero la soluzione del problema

$$\min ||c - yE||_2^2 \tag{2.20}$$

La regola **Y3**, conosciuta come la parte duale della regola di Mehotra, sceglie y come la soluzione duale del problema

$$\min \Big\{ cx + \tfrac{1}{2}x^T Qx + \tfrac{\tau_y}{2}(\|x\|^2 + \|u - x\|^2) : Ex = b \Big\}$$

ottenuto rilassando i vincoli di non negatività e imponendo i vincoli $Ex = b$. Dalle condizioni di ottimalità

$$c + Qx + \tau_x(2x - u) - yE = 0$$

$$Ex = b$$

ricaviamo infatti:

$$c + Q + 2x\tau_y - \tau_y u - yE = 0 \Leftrightarrow c + x(Q + 2\tau_y I) - \tau_y u - yE = 0$$

$$\Leftrightarrow x(Q + 2\tau_y I) = yE + \tau_y u - c \Leftrightarrow x = S(yE - c - \tau_y u),\ S = (Q + 2\tau_y I)^{-1}$$

sostituendo $x = S(yA - c - \tau_y u)$ in $Ex = b$ otteniamo

$$E(S(yE - c - \tau_y u)) = b \Leftrightarrow ESE^T y - cES\tau_y u = b \Leftrightarrow ESE^T y = b + ES(c - \tau_y u).$$

Una volta fissata y, indichiamo con $c^0 = c - yE + Qx$ il residuo dei vincoli duali che verrà impiegato in alcune formule d'inizializzazione delle altre variabili.

[6] veloci e poco raffinate

2.5.2 Inizializzazione variabili primali

Per le variabili primali x sono state proposte le seguenti regole:

X0) $x_i^0 := \begin{cases} \tau_x & se\ \tau_x \geq 0 \\ -\tau_x u_i & se -\frac{1}{2} \leq \tau_x < 0 \end{cases}$

X1) $x_i^0 := \begin{cases} \tau_x & se\ c_i^0 \geq 0 \\ u_i - \tau_x & altrimenti \end{cases}$

X2) $x^0 := SE^T(ESE^T)^{-1}b$

X3) $x^0 = S(E^T y^0 - c + \tau_y UB)$

X4) $x_i^0 := \begin{cases} \frac{u_i}{2} + \frac{\tau_x}{c_i^0} - k_i^0 & se\ c_i^0 > 0 \\ \frac{u_i}{2} + \frac{\tau_x}{c_i^0} + k_i^0 & se\ c_i^0 < 0 \\ \frac{u_i}{2} & se\ c_i^0 = 0 \end{cases}$, dove $k_i^0 = \sqrt{\left(\frac{u_i}{2}\right)^2 + \left(\frac{\tau_x}{c_i^0}\right)^2}$

La regola **X2** è la soluzione del problema

$$\min\left\{\ \tfrac{1}{2}x^T S^{-1}x : Ex = b\right\}$$

che equivale a

$$\min\left\{\ \tfrac{1}{2}x^T S^{-1}x + y(b - Ex)\right\}$$

le cui condizioni di ottimalità sono date da

$$Ex = b \tag{2.21}$$

$$S^{-1} - yE = 0. \tag{2.22}$$

da cui, operando i seguenti passaggi:

$$(2.21) \Longrightarrow b - Ex = 0; (2.22) \Longrightarrow x = SE^T y \Longrightarrow x - SE^T = 0 \tag{2.23}$$

$$(2.23) \Longrightarrow E(x - SE^T)^{-1}b \tag{2.24}$$

sostituendo 2.24 nell'equazione $Ex = b$ otteniamo così

$$x = SE^T(ESE^T)^{-1}b. \tag{2.25}$$

La regola **X3** è la primale della **Y3** [7], mentre la regola **X4** setta, a seconda del costo residuo $c^0(i)$, la variabile primale corrispondente più o meno in prossimità del lower bound, upper bound o nel mezzo del intervallo.

[7] Per la derivazione vedere i calcoli per la variabile duale **Y3**

2.5.3 Inizializzazione variabili di scarto duali

Per le variabili di slack duali (z, w) sono state proposte le seguenti regole

S0) i valori iniziali di w e z vengono scelti $\geq \tau_z$ in modo tale da soddisfare la relazione $y^0E + z^0 - w^0 = c$ o la relazione $y^0E - Qx^0 + z^0 - w^0 = c$ rispettivamente nel caso lineare o quadratico.

S1) $z_i^0 = \frac{\tau_z}{x_i^0}, \quad w_i^0 = \frac{\tau_z}{u_i - x_i^0}$

S2) $z_i^0 := \frac{\tau_z}{u_i} + \frac{c_i^0}{2} + \sqrt{\left(\frac{\tau_z}{u_i}\right)^2 + \left(\frac{c_i^0}{2}\right)^2}, \quad w_i^0 = \tau_z \frac{z_i^0}{u_i z_i^0 - \tau_z}$

S3) $z_i^0 = \frac{2\mu + c_i^0 u_i + \sqrt{4\mu^2 + (c_i^0 u_i)^2}}{2u_i}; \quad w_i^0 = \frac{2\mu - c_i^0 u_i + \sqrt{4\mu^2 + (c_i^0 u_i)^2}}{2u_i}$

La regola **S1** è un'inizializzazione "quick&dirty", mentre la **S2** è analoga alla **X4**. La regola **S3** deriva dalle condizioni KKT (2.17) eliminando l'equazione $Ex = b$

$$yE - w + z - Qx = c \Longrightarrow -w + z - Qx = c - yE \tag{2.26}$$

$$ZXE = \mu e \Longrightarrow x = \mu Z^{-1} e \tag{2.27}$$

$$(2.26, 2.27) \Longrightarrow w \;=\; z - Qx - c - yE \;=\; z - \mu Q Z^{-1} e - c - yE \tag{2.28}$$

Sia $DC = c - yE + w - z$

$$\Longrightarrow Zu - \mu e - \mu Q Z^{-1} u + \mu^2 Q Z^{-2} - DCu + \mu DC Z^{-1} e = \mu e \tag{2.29}$$

Sia $dc = c - yE$. Moltiplicando la (2.29) per Z^2, otteniamo

$$\Longrightarrow Z^3 u - Z^2(2\mu e + dc * u) + \mu Z(dc - Qu) + \mu^2 Q e = 0 \tag{2.30}$$

Fissando μ, in funzione di τ_z, si costruisce il relativo central path calcolando le radici dell'equazione di terzo grado (2.30)

$$z_i^0 = \frac{2\mu + c_i^0 u_i + \sqrt{4\mu^2 + (c_i^0 u_i)^2}}{2u_i},$$

$$w_i^0 = \frac{2\mu - c_i^0 u_i + \sqrt{4\mu^2 + (c_i^0 u_i)^2}}{2u_i},$$

$$x_i^0 = \frac{\mu}{z_i^0}$$

2.6 Direzione di Newton

Come la scelta delle soluzioni iniziali, anche la scelta della direzione dipende dal metodo che si utilizza. Nel metodo primale, si parte da una soluzione primale (x, s) e, approssimando linearmente le equazioni del sistema, si ottengono le variabili duali (y, z, w) corrispondenti alle direzioni $(\triangle_x, \triangle_s)$. Più precisamente dall'equazione $XZe = \mu e$ è possibile ottenere $z = \mu[X+\triangle X]^{-1}$ che, linearizzata mediante l'espansione di Taylor arrestata al prim'ordine, equivale a

$$z = \mu[X^{-1}e - X^{-2}\triangle X] \tag{2.31}$$

In maniera del tutto analoga è possibile ottenere $w = \mu[S+\triangle S]^{-1}e$ dall'equazione $WSe = \mu e$ che linearizzata equivale a

$$w = \mu[S^{-1}e - S^{-2}\triangle_x] \tag{2.32}$$

utilizzando queste due relazioni nelle restanti equazioni, è possibile ottenere esplicitamente le formule per il calcolo di y e$(\triangle_x, \triangle_s)$. In maniera alternativa è possibile derivare queste formule considerando la versione non lineare del problema primale (2.10) ottenuta eliminando i vincoli $0 \leq x \leq u$ ed inserendoli in funzione obiettivo per mezzo della funzione logaritmica. Sia quindi

$$\min \ \ (cx - \mu(\sum_i \ln x_i + \sum_i \ln s_i \ )) \tag{2.33}$$

$$\begin{gathered} Ex = b \\ x + s = u \end{gathered}$$

si costruisce il modello di secondo ordine rispetto ad (x, s)

$$\min \ (\ cx - \mu(X^{-1}\triangle_x + S^{-1}\triangle_s) + \frac{1}{2}\ \mu(\triangle_x X^{-2}\triangle_x + \ \triangle_s S^{-2}\triangle_s)) \tag{2.34}$$

$$\begin{gathered} E\triangle_x = b - Ex \\ \triangle_x + \triangle_s = u - x - s \end{gathered}$$

risolvendo il sistema KKT associato

$$\begin{gathered} E\triangle_x = b - Ex \\ \triangle_x + \triangle_s = u - x - s \\ yE - w = c - \mu(X^{-1}e - X^{-2}\triangle_x) \\ -w = -\mu(S^{-1}e - S^{-2}\triangle_s) \end{gathered} \tag{2.35}$$

si ottengono le formule per il calcolo di $\triangle_x, \triangle_s, z, w$.

$$\begin{array}{lccc} (E\theta E^T) & y & = E\theta(c + \mu(S^{-1} - X^{-1}e + S^{-1}(u - x - s))) + \mu(b - Ex) \\ & \triangle_x & = \frac{1}{\mu}\theta(y - Ec - \mu(S^{-1} - X^{-1}e + S^{-1}(u - x - s))) \\ & \triangle_y & = (u - x - s) - \triangle_x \\ & z & = \mu(X^{-1}e - X^{-2}\triangle X) \\ & w & = \mu(S^{-1}e - S^{-2}\triangle S) \end{array}$$

dove $\theta = X^2S^2(X^2+S^2)^{-1} = (X^{-2}S^{-2})^{-1}$.
Nel metodo duale, le variabili primali (x, s) corrispondenti alla direzione $(\triangle_y, \triangle_z, \triangle_w)$ sono derivate da un'appropriata linearizzazione delle equazioni del sistema KKT (2.15). Dalla quarta equazione del sistema (2.15) si ottiene

$$x = \mu[Z^{-1}e - Z^{-2}\triangle_z]$$

che sostituita nelle altre equazioni ci fornisce le formule esplicite per il calcolo di s e $(\triangle_y, \triangle_z, \triangle_w)$.

$$(E\theta E^T) \quad \begin{array}{cc} \triangle_y & = \frac{1}{\mu}b + E((c - yE - z + w) + (w - z) - (\frac{1}{\mu}W^2u) \\ x & = \mu(\theta E^T\triangle_y - ((c - yE - z + w) + (w - z) - (\frac{1}{\mu}W^2u) \\ \triangle_z & = z - \frac{1}{\mu}Z^2x \\ \triangle_w & = w - \frac{1}{\mu}W^2(u - x) \end{array}$$

dove $\theta = (Z^2+W^2)^{-1}$. In maniera alternativa, è possibile vedere queste formule come le formule per il passo di Newton della versione non lineare del problema duale (2.14), dove i vincoli $z \geq 0$ e $w \geq 0$ sono stati replicati dalla barriera logaritmica $-\mu(\sum_i ln(z_i) + \sum_i ln(w_i))$

$$\max(yb - wu + \mu(\sum_i ln(z_i) + \sum_i ln(w_i)) \tag{2.36}$$

$$yE + z - w = c$$

Costruendo infatti il modello del secondo ordine di (2.36) rispetto a (y, z, w) si ottiene

$$\max(\triangle_y b - \triangle_w u + \mu(Z^{-1}\triangle_z + W^{-1}\triangle_w) - \frac{1}{2}\mu(\triangle_z Z^{-2}\triangle_z + \triangle_w W^{-2}\triangle_w))$$

$$\triangle_y E + \triangle_z - \triangle_w = c - yE - z + w$$

il cui corrispondete sistema KKT è dato

$$\triangle_y E + \triangle_z - \triangle_w = c - yE - z - w$$

$$Ex = b$$

$$x = \mu(Z^{-1}e - Z^{-2}\triangle_z)$$

$$-x = -u + \mu(W^{-1}e - W^{-2}\triangle_w)$$

Infine, nella versione primale duale, si considera la coppia di (x, s) e (y, z, w) e ci si sposta lungo la direzione di Newton primale duale. Le formule per calcolarla, si ottengono riscrivendo le equazioni

$$SWe = \mu e$$
$$(x, s, z, w) \geq 0$$

del sistema KKT (2.17), come

$$(X + \triangle X)(Z + \triangle Z)e = \mu e$$
$$(S + \triangle S)(W + \triangle W)e = \mu e$$

e linearizzando il sistema non lineare ottenuto. Con opportuni calcoli algebrici le formule che si ottengono sono

$$DC = c - yE - z + w + Qx$$

$$v1 = (\mu I - ZX - \triangle Z \triangle X)e$$

$$v2 = (\mu I - SW - \triangle S \triangle W)e$$

$$r = \theta(S^{-1}(v2 - W(u - s - x)) - X^{-1}v1 + DC)$$

$$\begin{array}{lll} (E\theta E^T) & \triangle_y & = (b - Ex) + Er \\ & \triangle_x & = \theta E^T \triangle_y - r \\ & \triangle_s & = (u - s - x) - \triangle_x \\ & \triangle_w & = S^{-1}(v2 - W\triangle_s) \\ & \triangle_z & = DC + \triangle_w - E^T\triangle_y + Q\triangle_x \end{array}.$$

dove $\theta = WS^{-1} + ZX^{-1}$.

Osserviamo che le formule appena esposte non sono implementabili a causa della presenza dei termini non lineari $\triangle Z$, $\triangle X$ e $\triangle S$ nella definizione di $v1$ e $v2$. Per questo motivo nella versione primale duale standard questi termini vengono azzerati e quindi semplicemente ignorati. Nella versione Predictor-Corrector, invece, il sistema viene prima risolto ponendo $\mu = 0$, $\triangle_z = \triangle_x = \triangle_s = \triangle_w = 0$; dopodichè viene risolto altre $(Barrr - 1)$ volte, aggiornando μ automaticamente ad ogni iterazione, e impiegando i valori di $\triangle Z, \triangle X, \triangle S, \triangle W$ dell'iterazione precedente.

2.7 Aggiornamento del parametro barriera μ

Come affrontato al paragrafo 1.2, l'algoritmo necessita di un metodo che aggiorni dinamicamente il parametro barriera μ. Nel nostro software è possibile selezionare una tra le tre metodologie proposte:

- $\mu = \frac{\rho ZX}{2m}$
- $\mu = \frac{\rho ZX}{2m}$; $\rho = \frac{1-apD}{apD+10}^2$; $apD = \max XStep, ZStep$
- $\mu = \frac{ZX}{OldZX}^2 \frac{ZX}{2m}$

dove ρ è un parametro che serve ad aggiornare il valore di μ, m è il numero di colonne della matrice di incidenza E, ZX è il valore del duality gap, $OldZX$ è il valore del duality gap all'iterazione precedente, $XStep$ e $ZStep$ sono la dimensione dei passi nello spazio primale e duale rispettivamente. Scegliendo la prima formula è possibile inizializzare ρ, vedi 3.2.2, ad un valore a piacere o di default ($\rho = \frac{1}{2m}$).

Chapter 3

Analisi sperimentale

3.1 Descrizione del software

Il software è scritto interamente in `c++` ed è strutturato come mostrato in figura 3.1.

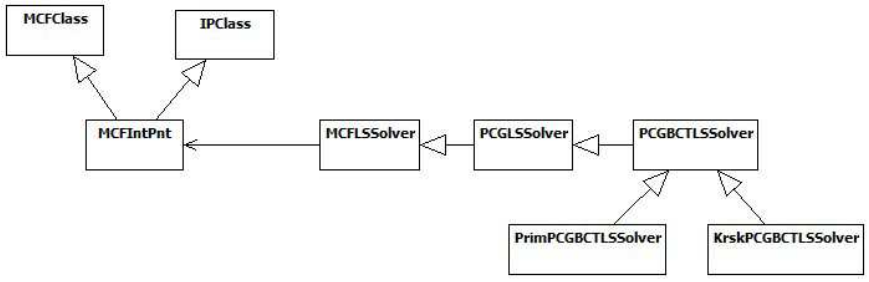

Figure 3.1: Diagramma UML delle classi del framework di lavoro.

- `IPClass` è una classe astratta [1] che implementa tutte le varianti dell'algoritmo IP discusse nel capitolo precedente, incluse tutte le regole di crash start. Trattandosi di una classe astratta, essa funge da scheletro per le classi derivate fornendo, solo virtualmente, l'accesso alla matrice dei coefficienti.

- `MCFClass` è una classe astratta che fornisce un'interfaccia standard per risolutori di problemi di flusso di costo minimo lineari e quadratici. I metodi pubblici della classe, correttamente ridefiniti nelle classi derivate, sono utilizzati per caricare l'istanza del problema, leggerne i parametri, risolverlo, raccoglierne i risultati, cambiare i costi/deficit/upper bounds,

[1] definisce la struttura di una gerarchia di classi, ma non può essere istanziata direttamente

e modificarne la struttura del grafo aggiungendo o rimuovendo archi e/o nodi.

- `MCFIntPnt` implementa un solutore di istanze del problema MCF, usando differenti varianti dell'algoritmo IP, e delegando la risoluzione del sistema KKT (1.9) ad un solutore generico di sistemi lineari, definito dalla classe `MCFLSSolver`.

- la classe astratta `MCFLSSolver` definisce un'interfaccia per la risoluzione del sistema di equazioni lineari posto nella forma

$$(E\theta E^T)\triangle y = d \tag{3.1}$$

- `PCGLSSolver` implementa un generico algoritmo del gradiente coniugato precondizionato per risolvere il sistema (3.1), utilizzando un semplice precodizionatore diagonale e lasciando la possibilità di implementarne di diversi e più sofisticati alle classi derivate.

- `PCGBCTLSSolver` è una classe astratta che deriva da `PCGLSSolver` ed implementa un generico precondizionatore tree/BCT+diagonale, lasciando alle sottoclassi il compito di implementare concretamente i metodi per il calcolo dell'albero di copertura e del BCT.

- `KrskPCGBCTLSSolver` implementa i metodi per il calcolo dell'albero di copertura e del BCT utilizzando un'euristica "tipo-Kruskal" basata su un ordinamento approssimato degli archi, selezionabile fra quelli proposti, o su una visita a ventaglio parametrizzata, [9].

- la classe `PrimPCGBCTLSSolver` è analoga alla `KrskPCGBCTLSSolver`, a meno dell'euristica impiegata, non più "tipo-Kruskal", ma "tipo-Prim", [9].

3.2 Testing

3.2.1 Istanze di test

Per le nostre prove abbiamo selezionato tre generatori ben noti di problemi MCF: `goto` (GridOnTorus), `gridgen` e `netgen` [10]. Per ognuno abbiamo prodotto quattro famiglie d'istanze denominate `genk_d`, dove `gen` è il generatore specifico, `n`=2^k è il numero di nodi, e `d` la densità. Ogni istanza è prodotta nel formato Dimacs standard [11].

La matrice dei costi quadratici Q viene calcolata a tempo d'esecuzione attraverso un generatore implementato nella classe `Main`. Più precisamente, quando si decide di voler trasformare il problema lineare in quadratico, è necessario fornire al software un omoniomo file aggiuntivo `<nomeistanza>.qdr` nel quale sono contenuti, nell'ordine, il seme[2] ed il parametro di scala $Qfactor$, che verranno

[2] valore iniziale dal quale generare i valori successivi

impiegati nella generazione dei costi quadratici. Settando opportunamente il parametro *QUADRATIC* della classe `Main`, è possibile generare costi pseudo-casuali a distribuzione uniforme nell'intervallo $[0, CQ]$, dove $CQ = 2\sqrt{\frac{\|C\|_1}{m}} * Qfactor$ o nell'intervallo $[0, QfactorC(i)]$, dove $C(i)$ è il costo dell'arco i-esimo. Si osservi che, intuitivamente, settare valori più alti al paramentro $Qfactor$ equivale ad aumentare il "peso" della parte quadratica; viceversa diminuirlo nè diminuisce il peso.

3.2.2 Studio dei parametri

Perchè l'applicazione funzioni è richiesto un apposito file, denominato `IPPar.bct`, che fornisce una sorta d'interfaccia per configurare vari parametri del solutore relativi sia all'algoritmo del punto interno che alla risoluzione del sistema KKT. Come sarà possibile evincere dai paragrafi successivi, larga parte del tirocino ha vertito sul tuning di una serie di questi parametri, dichiarati critici a causa della forte coorelazione tra il loro valore e l'efficacia della computazione. Il nostro lavoro si è focalizzato soprattutto sullo studio dei parametri del metodo del punto interno, testando solo superficialmente, per questioni di tempo fisico, la parte relativa alla scelta del precondizionatore nella risoluzione del sistema core. Di seguito è riportato l'elenco dei parametri relativi alla classe `IPClass` su cui abbiamo centrato la nostra attenzione.

- **WMthd** per scegliere se usare il metodo IP primale, duale o primale-duale
- **Barrr** per scegliere se utilizzare la versione dell'algoritmo affine o la standard. In particolare settare $Barrr = 0$ equivale ad usare la versione affine dell'algoritmo, in cui ad ogni iterazione si utilizzano le formule corrispondenti a mandare $\mu \to \infty$. Inoltre, nel metodo primale-duale, questo parametro controlla anche il numero di correzioni di centralità fatte per ogni iterazione. Cioè, se $Barrr = 1$ allora viene utilizzata la versione standard primale-duale affine, mentre se $Barrr > 1$ viene impiegata la versione Predictor Corrector in cui ad ogni iterazione vengono effettuate $(1 - Barrr)$ correzioni.
- **MaxIt** per settare il numero massimo di iterazioni dell'algoritmo IP
- **NwtnS** per scegliere fra imporre i passi del primale e del duale uguali ($NwtS \in 2, 3$) o diversi ($NwtS \in 0, 1$) e maggiori ($NwtS \in 0, 2$)o minori uguali ($NwtS \in 1, 3$)ad uno. Settando
- **Malpha** per assicurare che il prossimo punto sia "abbastanza interno", la grandezza del passo viene moltiplicata per un fattore < 1. *Malpha* rappresenta il valore massimo o il valore che può essere assunto da questo fattore.
- **Valpha** per calcolare dinamicamente il fattore di ammissibilità.
- **Palpha** fattore ammissibilità per le iterazioni del metodo Predictor-Corrector.

- **IntMu** per inizializzare il moltiplicatore della funzione barriera ad un valore a piacere o ad uno predefinito.
- **IntRho** per inizializzare il parametro che viene impiegato ad ogni iterazione per aggiornare il valore di Mu, vedi paragrafo 2.7.
- **OptEps** precisione relativa richiesta per il soddisfacimento delle condizioni di ottimalità
- **FsbEps** precisione relativa richiesta per il soddisfacimento dei vincoli.
- **PosEsp** per imporre che tutte le variabili non negative siano maggiori o uguali a questo parametro.
- **SysEps** per impostare la precisione relativa richiesta per la risoluzione del sistema core, calcolata come il prodotto tra *SysEps* e *FsbEps*. Nei metodi primale e primale duale viene utilizzata, nel caso in cui non sia ancora stata raggiunta l'ammissibilità primale) il prodotto tra *SysEps* e la precisione relativa corrente raggiunta nell'ammissibilità primale.
- **CnvEps** utilizzato insieme a *CnvStP* per testare la convergenza. L'inammissibilità primale/duale, il duality gap, il valore della funzione obiettivo primale/duale cresce/decresce linearmente ad ogni iterazione di almeno *CnvEps* per un periodo di *CnvStp* passi consecutivi.
- **CnvStp** vedi descrizione parametro *CnvEps*. Settare $CnvStp = 0$ equivale ad applicare una riduzione lineare, mentre impostare $CnvStp > 0$ potrebbe provocare uno stallo temporaneo della convergenza, potenzialmente causato dalla risoluzione inesatta del sistema core.
- **InitX** per selezionare una fra le formule viste al paragrafo 2.5.2 per inizializzare la soluzione primale
- **XThrsh** corrisponde al paramentro τ_x del paragrafo 2.5.2
- **InitY** per selezionare una fra le formule viste al paragrafo 2.5.1 per inizializzare la soluzione duale
- **YThrsh** corrisponde al paramentro τ_y del paragrafo 2.5.1
- **InitZ**per selezionare una fra le formule viste al paragrafo 2.5.3
- **ZThrsh** corrisponde al paramentro τ_z del paragrafo 2.5.3

In realtà, come anticipato, dal file IPPar è possibile impostare anche i parametri degli altri moduli che compongono il software e che permettono di configurare l'algoritmo del gradiente coniugato precondizionato e selezionare il precondizionatore. Di questi, è stato considerato solo il parametro relativo alla scelta del precondizionatore, al fine di identificarne almeno uno efficace nella risoluzione del sistema KKT. Nella nostra analisi, abbiamo utilizzato la variante primale-duale con barriera logaritmica ed abbiamo mantenuto la configurazione funzionante

per la versione del software lineare per quanto concerne i valori della precisione per l'ottimalità, precisione per l'ammissibilità, minimo numero positivo, precisione per sistema core, soglia convergenza e numero massimo iterazioni non convergenti. Allo stesso modo, sono state sfruttate le informazioni derivanti dallo studio della versione lineare, per estrapolare la configurazione dei moduli non considerati.

3.2.3 Strumentazione utilizzata

I test sono stati effettuati su piattaforma Windows in più macchine equivalenti del laboratorio del dipartimento [3]. Per la raccolta e l'elaborazione dei dati è stato utilizzato Microsoft Excel.

3.2.4 Test preliminari

Nella prima fase di sperimentazione è stata valutata la correttezza del software al caso lineare. Per farlo è stato utilizzato `MCFSimplex`[12], un'implementazione del simplesso primale-duale specifico per il problema MCF, ed abbiamo semplicemente verificato su un numero significativo di istanze la corrispondenza tra gli input. Già da questa fase è stato possibile osservare il fondamentale ruolo di alcuni parametri nell'efficacia e nell'efficienza della computazione. Dopo averne selezionato un gruppo "critico" (formule di crash start, interior factor, precondizionatore), ne abbiamo effettuato un piccolo tuning al fine di trovare una combinazione tale da rendere il software efficace.

Guidati dalla base teorica, abbiamo affrontato per primo lo studio del parametro di ammissibilità e quindi di `MAlpha` e `VAlpha`. Specificatamente, individuata una formula funzionante, abbiamo valutato sul campione di istanze la convenienza nell'effettuare passi lunghi, vicini all'unità, piuttosto che brevi, vicini allo zero, lungo la direzione di ricerca. Per l'analisi, abbiamo effettuato test specifici modificando opportunamente il valore dei parametri in gioco ed abbiamo osservato il loro ruolo nell'avanzamento dell'algoritmo. La tabella 3.1 riassume i risulati raccolti per la combinazione di crash start X1+Y1+S3, fissando i passi del primale e duale potenzialmente diversi, e minori uguali ad 1. Dall'analisi dei dati è stato possibile osservare la convenienza nell'effettuare passi lunghi (`Malpha` = 0.99, 0.999, 0.9995) anzichè brevi e nel tralasciare un aggiornamento dinamico del parametro (`Valpha` = 1).

Fissato dunque questo gruppo di valori, abbiamo studiato la lunghezza del passo nello spazio del primale e del duale. Da quest'indagine si evince che settare `Malpha` ad un valore nell'insieme 0.99, 0.999, 0.9995 è indifferente dal punto di vista del numero di iterazioni, analogamente scegliere se avere passi minori uguali o maggiori di uno. Il miglioramento lo si ottiene scegliendo passi potenzialmente diversi (`NwtS` $\in 0, 1$).

L'indagine è proseguita valutando l'incidenza delle soluzioni di partenza ovvero

[3] AMD Phenom (tm) IIX x4 945 Processor, 3000 Mhz, 4 core, 4 processori logici, 8 Gb RAM

MAlpha	VAlpha	Numero Medio Iterazioni
0.5	0	75.4
	1	74.4
	0.5	75.4
0.9	0	44.6
	1	43.6
	0.5	46
0.99	0	42.4
	1	41.4
	0.5	44.2
0.999	0	42.4
	1	41.4
	0.5	44.2
0.9995	0	42.4
	1	41.4
	0.5	44.2

Table 3.1: studio della dimensione del passo lungo la direzione di ricerca per le istanze lineari

	grid14_8		net14_8		goto14_8	
Precondizionatore	it_i	t_i	it_i	t_i	it_i	t_i
9	41.4	43.8	46	171.9	46.2	9.73
13	41.4	7.5	45.4	11.0	46.0	86.1

Table 3.2: incidenza del precondizionatore nella risoluzione delle istanze lineari

stimando se, inizializzare l'algoritmo con una combinazione di formule di crash start piuttosto che un'altra potesse migliorarne, peggiorarne o, addirittura abbatterne, la convergenza. Come ragionevolmente ci aspettavamo, a diversi punti di partenza corrispondono diversi comportamenti. Gli scenari possibili sono, in sostanza, tre:

- la procedura trova una soluzione primale/duale in numero di iterazioni finito e in un tempo misurato in secondi
- la procedura non converge perchè non sta effettivamente progredendo verso la soluzione ottima
- la procedura non converge a causa di errori numerici riscontrati nella risoluzione del sistema KKT

Dal momento che il primo caso corrisponde evidentemente al caso ottimo, soffermiamoci alle altre due eventualità. Il punto focale consiste nel capire se la combinazione di formule che si vaglia fallisce perchè rappresenta effettivamente

	grid14_8		net14_8		goto14_8	
Precondizionatore	it_i	t_i	it_i	t_i	it_i	t_i
9	50.8	148.2	73.8	3116.1	43.2	7.1
13	50.8	11.3	54.6	432.8	43	595.9

Table 3.3: incidenza del precondizionatore nella risoluzione delle istanze quadratiche

un cattivo punto di partenza oppure se si tratta di un "fallimento apparente" ovvero se, modificando opportunamente altri parametri in gioco, sia possibile renderla robusta. In questo contesto è entrato in gioco l'importante funzione del precondizionatore. Gli errori che vengono restituiti dal solutore sono infatti di natura diversa. Il fatto che il solutore non riesca a convergere verso l'ottimo perchè sta facendo passi di miglioramento troppo piccoli equivale ad un problema nell'algorimo Interior Point e quindi potenzialmente, posto che gli altri parametri elencati 3.2.2 siano settati correttamente, della combinazione di crash start scelta. D'altro canto, il fatto che sussistano errori numerici nella risoluzione del sistema core è sintomatico di una problematica nel solutore KKT e quindi potenzialmente di una cattiva scelta del precondizionatore. Risulta chiaro che non è possibile trattare ogni parametro come a sè stante, ma come un tassello di una configurazione più ampia. Non solo. Sperimentando la coppia <combinazione di crash start, precondizionatore> è stato possibile evidenziare una dipendenza tra tipologia di rete e precondizionatore. Con reti di tipo netgen e gridgen si ottengono risultati migliori con precondizionatori che sfruttano il residuo diagonale, viceversa per le grid on torus. Nella tabella 3.2.4 mettiamo in evidenza questo fatto per la combinazione di crash start X1+Y1+S3.

3.2.5 Test sulle istanze quadratiche

Acquisita una certa familiarità col software, ne abbiamo sperimentato l'efficacia in presenza di archi con costo quadratico. Per valutare il caso medio, abbiamo assunto $Qfactor = 1$ ed abbiamo così generato i costi quadratici per le istanze precedenti. L'analisi dei dati mostra come, analogamente al caso lineare, anche in quello quadratico sia possibile evidenziare una relazione fra precondizionatore e tipologia di rete. Come è infatti possibile osservare dalla tabella 3.2.5 , mentre per le `goto` si ottengono migliori risultati settando il precondizionatore ad albero con algoritmo di tipo MST (Precondizionatore = 9) , per le `grid` e le `net` è più conveniente utilizzare lo stesso precondizionatore con l'aggiunta del residuo diagonale (Precondizionatore = 13). Alla luce di questo fatto, abbiamo raccolto i dati solo per il precondizionatore reputato migliore.

Decretato il precondizionatore, la fase successiva consiste nel selezionare un set di combinazioni di formule di crash start robusto, ovvero una set di combinazioni che funzionino, non necessariamente benissimo, in tutte le istanze. Per cercarla, siamo partiti da una formula che funzionava bene al caso lineare ed abbiamo

		grid14_8			net14_8			goto14_8		
NwtS	MAlpha	it_i	t_i	eff	it_i	t_i	eff	it_i	t_i	eff
0	0.99	41.4	7.6	100%	81.6	314.9	0%	72	586.9	0%
	0.999	41.4	6.9	100%	79.2	615.5	0%	81.4	81.4	0%
	0.9995	41.4	7.7	100%	74.8	405.2	0%	81.4	505.4	0%
1	0.99	41.4	7.7	100%	81.6	281.4	0%	67.2	17.5	40%
	0.999	41.4	7.5	100%	79.2	276.2	0%	66.4	14.5	40%
	0.9995	41.4	7.5	100%	74.8	279.5	0%	59.2	15.5	40%
2	0.99	45.4	8.4	100%	54.6	480.5	100%	43.2	15.5	100%
	0.999	45.4	8.5	100%	54.6	515.6	100%	43.2	13.0	100%
	0.9995	45.4	8.7	100%	54.6	458.0	100%	43.2	14.2	100%
3	0.99	45.4	7.6	100%	54.6	502.1	100%	43.6	497.2	100%
	0.999	45.4	9.3	100%	54.6	491.0	100%	43.4	406.9	100%
	0.9995	45.4	8.1	100%	54.6	490.2	100%	43.0	410.5	100%

Table 3.4: studio della grandezza del passo nello spazio del primale e del duale per istanze quadratiche

	grid				net		goto			
	14_8	**14_64**	**16_8**	**16_64**	**14_8**	**14_64**	**14_8**	**14_64**	**16_8**	**16_64**
X0+Y2+S3	20.6	69.3	165.1	444.9	561.2	891.3	7.9	331.0	171.6	7377.1
X0+Y0+S3	17.2	56.3	167.7	462.4	465.0	1182.6	17.3	471.6	171.8	9549.8
X3+Y2+S3	12.8	56.2	121.5	310.2	329.7	556.2	7.9	985.5	171.1	9458.8
X2+Y1+S0	8.1	40.9	113.0	344.4	657.9	4363.3	15.4	402.5	318.2	10164.4
X2+Y2+S3	12.8	56.3	121.8	310.9	329.8	553.2	8.0	332.8	171.9	9434.7

Table 3.5: set formule robuste nella risoluzione delle istanze quadratiche

iniziato a testarla in tutte le istanze, e, una volta ottenuto il responso, a variarla mano a mano sempre di più testando ogni volta la variante ottenuta. La sperimentazione ha nuovamente mostrato, come la tipologia di rete influisca sulla scelta delle formule d'inizializzazione. In particolare, mentre le `grid` e le `net` risultano risolvibili, più o meno efficientemente, con all'incirca tutte le formule, le `goto` sono solubili solo con determinate combinazioni di formule.

La ricerca è proseguita incrementando il "peso" della parte quadratica, ponendo $Qfactor = 100$, e mantendo invariata la configurazione dei parametri. A questo livello abbiamo compiuto l'osservazione più rilevante del nostro studio. La convergenza dei metodi del punto interno è dimostrata nella teoria per passi nel primale e nel duale uguali; nella pratica, ovvero sino a questo punto della sperimentazione, avevamo omesso questo particolare senza riscontrare alcun problema. In realtà, aumentando *Qfactor*, passando dalla programmazione lineare alla quadratica, questa proprietà assume un'importanza diversa e, perchè il metodo converga, risulta necessario porre uguali i due passi [13].

La tabella 3.4 mostra i dati i raccolti per la formula X1+Y1+S3 per i campioni d'istanze delle famiglie `grid`14_8, `net`14_8 e `goto`14_8. Nella colonna it_i troviamo il numero d'iterazioni medio che serve per risolvere un'istanza, nella colonna t_i troviamo il tempo medio, misurato in secondi, per risolvere un'istanza, ed infine nella colonna eff viene misurata la probabilità di risoluzione di quella parti-

colare famiglia d'istanze con quel particolare settaggio calcolata come il rapporto tra il numero di istanze risolte sul numero di istanze da risolvere. Come volevasi dimostrare, le istanze risultano risolvibili ponendo i passi del primale e del duale uguali ($NwtS \in 2, 3$).

Dal momento che il numero di iterazioni rimane invariato sia per passi minori uguali che maggiori di uno, abbiamo optato per settare $NwtS = 2$ perchè migliore dal punto di vista del tempo. Per quanto concerne *Malpha*, esso è stato settato, sempre per ragioni temporali, uguale a 0.9995. In particolare, dal momento che non era nostro scopo specializzarci in una sola classe di istanze, ci siamo fatti guidare dalla famiglia che necessita più tempo per essere risolta, ed abbiamo scelto *Malpha* come il minimizzatore del tempo medio impiegato a risolvere un'istanza delle `net`. In questo modo, è stato possibile raggiungere l'obiettivo del nostro lavoro, ovvero trovare un metodo in grado di risolvere con il metodo del punto interno le istanze quadratiche del problema di flusso di costo minimo.

Per valutare l'incidenza del punto di partenza e trovare un insieme di configurazioni di inizializzazione robuste, la ricerca è progredita valutando le varie combinazioni disponibili. Di queste, alcune sono state scartate poichè valutate cattivo punto di partenza, mentre altre hanno concorso a diventare le candidate a risolvere più efficientemente il problema. La tabella 3.2.5 raccoglie il set di formule robusto che è stato identificato.

Si osservi che le combinazioni che non compaiono non sono state inserite perchè considerate punti di partenza sterili, ovvero o non in grado di portare l'algoritmo ad una soluzione ottima o associate a tempi di risoluzione estremamente lunghi da renderle inutilizzabili. In base alla mole di tempo di risoluzione richiesta per risolvere la batteria di istanze in esame osserviamo per mezzo del grafico 3.2 che, per ottenere risultati migliori in termini di tempo, è bene lavorare con abbinando la Y2 alla S3, in particolare la formula di crash start che funziona

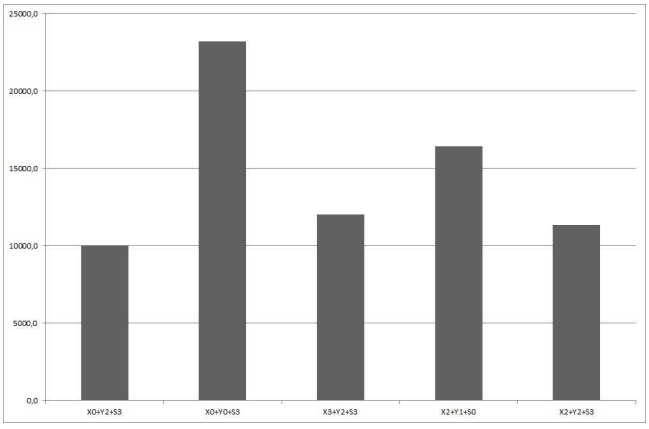

Figure 3.2: Valutazione bontà del punto di partenza

	0.01		1		100	
	t	**it**	**t**	**it**	**t**	**it**
grid14_8	8.8	47.4	8.2	48.0	17.2	50
grid14_64	42.6	51.2	44.6	52.0	56.3	52.6
grid16_8	54.3	54.4	126.4	50.6	171.8	50.8
grid16_64	623.2	60.8	355.3	57.6	462.5	59.4
goto14_8	9.0	40.6	11.3	43.0	17.3	43.4
goto14_64	193.3	77.6	270.1	76.4	471.6	81.8
goto16_8	50.8	47.8	126.4	50.6	171.8	50.8
goto16_64	1959.0	93.4	4185.3	97.2	9549.7	104.2
net14_8	15.6	46.6	35.9	51.8	464.9	51.4
net14_64	124.4	69.8	1821.1	77.3	766.4	78.6

Table 3.6: comportamento del software in relazione alla dimensione della parte quadratica

meglio è la X0+Y2+S3. Limitandoci invece ad una sola famiglia di istanze, osserviamo invece che è possibile trovare altre formule migliori. Se ad esempio, ci limitiamo a considerare le reti `grid14_8`, `grid14_64`, `grid16_8` e controlliamo i valori contenuti nella tabella 3.2.5 osserviamo il punto di partenza reputato migliore risulta essere quello associato alla combinazione X2+Y1+S0. Ne deduciamo che la bontà di un punto di partenza non è universale per ogni tipologia di rete, ma dipende dal dominio sul quale è applicato: in linea di principio, infatti, la X0+Y2+S3 funziona bene, ma non è detto che, limitandosi ad una sola tipologia di famiglia, non sia possibile trovarne altre di migliori.

Per valutare il comportamento del software in relazione alla dimensione della parte quadratica, i test sono stati effettuati settando $Qfactor$ a 0.01, 1, 100. La tabella 3.2.5 contiene i dati raccolti per la combinazione X0+Y0+S3 E' interessante notare come il numero delle iterazioni dell'algoritmo interior point non cresca proporzionalmente all'aumentare della dimensione della parte quadratica, ma anzi rimanga pressochè costante, vedi grafici 3.3, 3.4, 3.5.

3.3 Performance del metodo

Nel campo della ricerca operativa, il più importante elemento da considerare per determinare la bontà di un qualsiasi solutore è rappresentato dall'efficacia con cui esso riesce ad ottimizzare la funzione di costo. Identificata quindi una parametrizzazione del software consistente, ci siamo introdotti nel miglioramento della stessa, allo scopo di trovare un modo in grado di fornire performance migliori delle preesistenti. Abbiamo dunque indagato a proposito dei parametri della funzione barriera μ e ρ, sinora settati uguali a 0 e 0.5, rispettivamente. I valori testati ed i corrispondenti valori sono contenuti all'interno della tabella 3.3. Osserviamo che la scelta compiuta sinora funziona abbastanza bene, ma

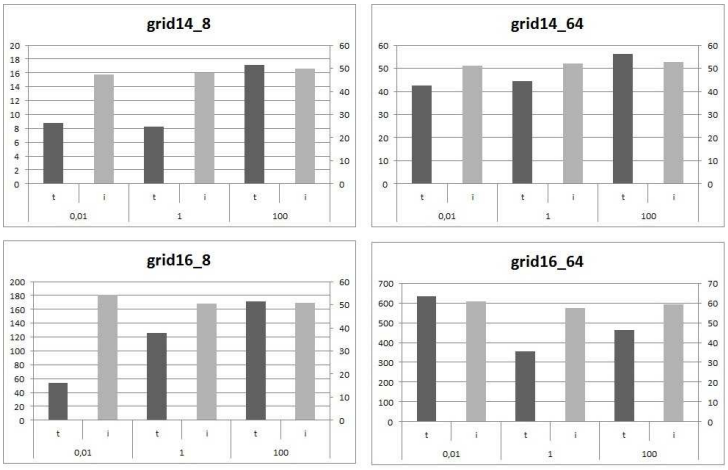

Figure 3.3: analisi comportamento in relazione a *Qfactor*

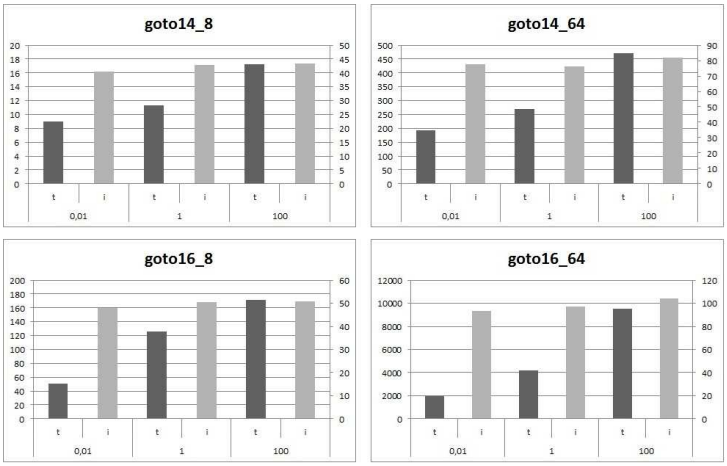

Figure 3.4: analisi comportamento in relazione a *Qfactor*

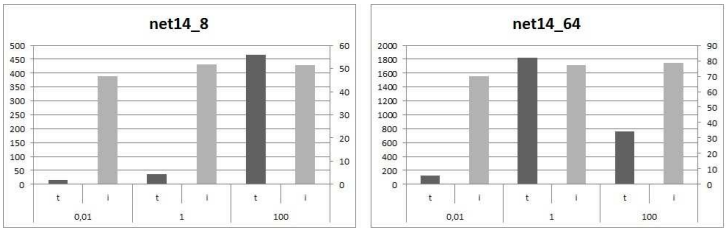

Figure 3.5: analisi comportamento in relazione a *Qfactor*

		grid14_8		net14_8		goto14_8	
μ	ρ	**it**	**t**	**it**	**t**	**it**	**t**
0	0	64	16.5	92.2	48.9	58.8 [*]	1572.02
	0.3	39.2	15.3	45.0	110.1	39.6	269.0
	0.5	50.8	17.9	54.6	458.0	43.2	14.3
	0.8	140.2	50.1	141.0	1009.7	85.2	632.5
0.5	0	64	18.1	96.0	91.1	59.4	442.5[*]
	0.3	58.2	12.3	45.0	237.6	39.6	494.4
	0.5	50	19.4	54.6	693.5	43.6	1564.0
	0.8	134.0	48.0	137.0	2019.4	84.2	1260.18

Table 3.7: studio delle performance

potenzialmente se ne potrebbero ottenere di migliori diminuendo il valore di ρ. Al contario, scegliere un valore di ρ vicino ad uno non porterebbe alcun beneficio, mentre automatizzarlo al valore di default, compiendo la scelta $\rho = 0$, non solo porterebbe ad un peggioramento nell'efficienza, ma comprometterebbe anche l'efficacia del solutore [4].

[4] i simboli [*] nella tabella 3.3 indicano che l'algoritmo ha terminato restituendo un errore

Chapter 4

Conclusioni

4.1 Difficoltà incontrate

Nel corso di questo tirocinio le difficoltà maggiori sono derivate dal doversi scontrare per la prima volta con metodologie di risoluzione alternative al simplesso e nel doversi confrontare con un complesso codice preesistente. Per riuscire a lavorare su questo progetto è stato necessario compiere una serie di studi preparatori atti a colmare le lacune esistenti riguardanti i metodi del punto interno applicati alla programmazione lineare e non, e di tutte le nozioni teoriche che lo sottointendono. Per quanto concerne la fase di test, il maggiore ostacolo è dipeso dalla mancanza di esperienza nel contesto e quindi nel non possedere a priori tutti gli strumenti per affrontare le problematiche che sono state riscontrate in corso d'opera, ma nell'imparare a costruirseli mano a mano sulla base dei riscontri pratici.

4.2 Valutazione critica del lavoro svolto

L'obiettivo principale del tirocinio consisteva nell'estensione al caso quadratico di un codice interior point preesistente per il problema di flusso di costo minimo lineare, con relativa fase di testing e tuning dei parametri algoritmici. Alla luce del lavoro svolto, possiamo considerare raggiunto ciò che ci eravamo prefissi. Tuttavia, come in ogni ricerca scientifica, l'indagine è ancora aperta. Sarebbe interessante, ad esempio, studiare il comportamento del software cambiando il metodo per il calcolo di copertura sostituendo l'euristica di Kruskal con quella di Prim. Inoltre in questo lavoro ci siamo soffermati solo su alcuni parametri del file di configurazione, tralasciandone altri che, potenzialmente, potrebbero migliorare le performance del nostro solutore: come visto al paragarafo 3.3, infatti, esistono metodi con cui è possibile operare una fase di perfezionamento. A causa di tempi fisici piuttosto lunghi e di una strumentazione inadeguata, il nostro contributo è stato fornire un'applicazione specializzata per il problema di flusso minimo che funzioni anche in presenza di archi di costo quadratico, senza

dare alcuna garanzia sulle prestazioni, ma proponendo vie di miglioramento. A fronte di questo fatto, è nostra intenzione approfondire questo aspetto in un secondo momento.

Appendix A

Ottimizzazione Non Vincolata

A.1 Introduzione

Come anticipato nella prefazione, molte applicazioni del mondo reale possono essere formulate come problemi di ottimizzazione. Formalmente l'ottimizzazione consiste nel determinare il valore di un vettore di variabili di decisione $x \in \mathbb{R}^n$ che massimizza o minimizza una funzione $f : \mathbb{R}^n \to \mathbb{R}$, quando x è vincolato ad appartenere ad un insieme ammissibile $\mathcal{F} \subseteq \mathbb{R}^n$

$$\min \left\{ f(x) \ : \ x \in \mathcal{F} \right\} \tag{A.1}$$

Si osservi immediatamente che la formulazione appena data non contraddice l'esposizione precedente in quanto un problema di minimo può essere sempre ricondotto ad uno di minimo, semplicemente cambiando di segno la funzione obiettivo.

$$\max f(x) = -\min(-f(x))$$

In base a questa puntualizzazione, sarà possibile, senza perdità di generalità, riferirsi nel corso della trattazione a problemi di minimizzione. E' appropriato notare immediatamente che, assegnata la funzione obiettivo e la regione ammissibile, potrebberero non esistere soluzioni ottime. Questo potrebbe essere dovuto alla non esistenza di punti ammissibili, ovvero all'esistenza di una regione ammissibile vuota; oppure alla non esistenza di un valore minimo della funzione sull'insieme $\mathcal{F}$. Quindi, solo nel caso cui esistono punti di minimo globale di $f$su $\mathcal{F}$ ci si può porre nel problema della ricerca di una soluzione ottima. A seconda di come è fatta la regione ammissibile è possibille classificare i problemi di ottimizzazione in due grandi famiglie. Nel caso in cui $\mathcal{F} \equiv \mathbb{R}^n$si parla di ottimizzazione non vincolata, mentre quando $\mathcal{F} \subset \mathbb{R}^n$, cioè è un sottoinsieme proprio di $\mathbb{R}^n$, si parla di ottimizzazione vincolata, ed in questo caso $\mathcal{F}$ è descritto da vincoli sulle variabili di decisione. I vincoli possono essere di disuguaglianza o di

uguaglianza. In entrambi i casi si tratta di funzioni scalari di x che definiscono, rispettivamente, delle disequazioni e delle equazioni che il vettore delle variabili deve soddisfare. Nel corso dell'esposizione, utilizzaremo la notazione $h(x)$ per riferirci ai vincoli di uguaglianza e $g(x)$per riferirci a quelli di disuguaglianza. Quando almeno una delle funzioni $f(x), h(x), g(x)$ è non lineare, rispetto ad almeno una delle componenti del vettore , si parla di Programmazione Non Lineare (PNL), viceversa se tutte le funzioni sono combinazioni lineari delle variabili di decisione allora si parla di Programmazione Lineare (PL). Si osservi che un problema di PL è necessariamente vincolato, poich altrimenti si tratterebbe sempre di un problema illimitato. è inoltre importante osservare che i problemi di PNL costituiscono una generalizzazione dei problemi di PL. In generale, non è possibile identificare per via analitica le soluzioni ottime di un problema di PNL, tuttavia le proprietà analitiche del problema possono tornare utili per valutare l'ottimalità di una soluzione in qualche modo generata. Scopo dell'ottimizzazione è quindi sviluppare algoritmi che permettano la risoluzione del problema, ossia che determinino un punto di minimo globale della funzione obiettivo sull'insieme ammissibile.

Definizione 2 (Punto di minimo globale) *Un punto $x^* \in \mathcal{F}$ è un punto di minimo globale di f su $\mathcal{F}$ se*

$$f(x^*) \leq f(x) \quad \forall x \in \mathcal{F}$$

e in tal caso, si dice che $f(x^*)$ è il valore minimo, o il minimo, globale f su F.

Gli algoritmi di ottimizzazione sono sequenziali: a partire da una soluzione iniziale x_0, non necessariamente ammissibile, creano iterativamente una sequenza di vettori $x_1, x_2, , x_k$ che sono tutti, sperabilmente, approssimazioni migliori di una soluzione ottima. La strategia adottata per generare la serie di soluzioni distingue i vari algoritmi. La maggior parte di essi utilizza i valori delle funzioni in gioco f, g, h e, all'occorrenza, anche i valori delle loro derivate prime e seconde; altri usano le informazioni raccolte durante il processo iterativo di ricerca; ed altri ancora usano solo l'informazione locale della soluzione corrente. In ogni caso, comunque, si vogliono determinare algoritmi efficienti, in grado di garantire una soluzione del problema in tempi di calcolo ragionevoli; robusti, in grado di risolvere, con prestazioni paragonabili, diverse instanze dello stesso problema e accurati ovvero non troppo sensibili ad errori nei dati o ad inevitabili errori di arrotondamento in cui ci si imbatte nell'implementazione. Purtroppo queste caratteristiche sono spesso conflittuali, e risulta necessario scendere a compromessi.

A.2 Problemi di ottimizzazione convessi

Un concetto che riveste molta importanza in ottimizzazione è quello di convessità. I problemi che, infatti, possiedono questa proprietà sono in genere più facili da risolvere, non solo in teoria, ma anche nella pratica.

Definizione 3 (Insieme convesso) *Un insieme $X \in \mathbb{R}^n$ è convesso se presi comunque due punti $x, y \in X$, allora il segmento che li congiunge è tutto contenuto in X, ossia se*

$$x, y \in X, \quad \lambda \in \mathbb{R}, \quad 0 \leq \lambda \leq 1 \Rightarrow (1-\lambda)x + \lambda y \in X.$$

Si consideri ora un insieme convesso X, e una funzione f definita su tale insieme.

Definizione 4 (Funzione convessa) *Una funzione $f : \mathbb{R}^n\mathbb{R}$ è convessa se il suo dominio è un insieme convesso $X \subseteq \mathbb{R}^n$ e comunque presi due punti $x, y \in X$*

Definizione 5 (Problema convesso) *Si dice che il problema (P) è un problema di ottimizzazione convesso se l'insieme ammissibile $\mathcal{F}$è un insieme convesso e la funzione obiettivo $f(x)$ è una funzione convessa su $\mathcal{F}$.*

Fortunatamente la grande maggioranza dei problemi di ottimizzazione godono di questa caratteristica che induce ad alcune proprietà che ne semplificano l'analisi e la risoluzione.

Teorema 6 (Assenza ottimi locali) *Un problema di ottimizzazione convesso o non ha soluzione, o ha solo soluzioni globali; non può avere soluzioni esclusivamente locali.*

è chiaro che stabilire se un problema è convesso è un passo fondamentale nella risoluzione di un problema di ottimizzazione. Per dichiarare che il problema è convesso, è necessario verificare che lo siano la funzione obiettivo e la regione ammissibile. Sfruttando le proprietà relative agli insiemi e alle funzioni convesse, procediamo ad esporre e dimostare le condizioni di convessità di un problema di programmazione non lineare.

Proprietà 7 (Insiemi convessi) *Siano X ed Y due insiemi convessi ed $\alpha > 0$ uno scalare, allora*

1. *l'insieme αX è convesso*
2. *l'insieme $X + Y$ è convesso*
3. *l'insieme $X \cap Y$ è convesso*

Proprietà 8 (Funzioni convesse) *Siano f e g due funzioni convesse ed $\alpha > 0$ uno scalare, allora*

1. *la funzione αf è convessa*
2. *la combinazione lineare $f + g$ è convessa*
3. *la funzione $\max f, g$ è convessa*
4. *il luogo dei punti di x per i quali vale $f(x) < \alpha$ è convesso*

Teorema 9 (Condizioni di convessità di un problema di PNL) *Un problema di programmazione non lineare (PNL)*

$$\min\left\{f(x) \ : \ g_i \leq 0, \ i = 1, ..., k \quad h_j \leq 0, \ j = 1, ..., w\right\} \tag{A.2}$$

è convesso se $f(x)$ è convessa, le g_i sono convesse e le h_j sono lineari.

Se è f è una funzione convessa differenziabile, è possibile dare le condizioni necessarie e sufficienti di convessità espresse attraverso l'uso delle derivate della funzione. Nei teoremi successivi sono riportati, rispettivamente, i risultati relativi alle condizioni espresse per mezzo delle derivate prime e seconde di f.

Teorema 10 (Condizioni necessarie e sufficienti di convessità) *Sia $\mathcal{F}$ un insieme convesso aperto, e sia $f : F \to \mathbb{R}$. Sia inoltre ∇f continuo su $\mathcal{F}$. Allora f è convessa su $\mathcal{F}$ se e solo se, per tutte le coppie di punti $y, z \in \mathcal{F}$ si ha*

$$f(y) \geq f(z) + \nabla f(x)(y - z).$$

Geometricamente, la condizione appena esposta esprime il fatto che una funzione è convessa su $\mathcal{F}$ se e solo se in un qualsiasi punto $y \in \mathcal{F}$ l'ordinata $f(y)$ della funzione non è inferiore alle ordinate dei punti del piano tangenti al grafo della funzione in un qualsiasi altro punto $z \in \mathcal{F}$.

Teorema 11 (Condizioni necessarie e sufficienti di convessità) *Sia F un insieme convesso aperto, e sia $f : \mathcal{F} \to \mathbb{R}$. Sia inoltre la matrice Hessiana $\nabla^2 f$ continua su $\mathcal{F}$. Allora f è convessa su $\mathcal{F}$ se e solo se, per ogni $x \in \mathcal{F}$ la matrice $\nabla^2 f(x)$ è semidefinita positiva.Se inoltre la matrice Hessiana è anche definita positiva, allora è strettamente convessa.*

A.3 Condizioni analitiche di ottimalità

La prima questione che occorre risolvere nel momento in cui si decide di risolvere un qualsiasi problema di ottimizzazione è la caratterizzazione dei punti di ottimo, o in altri termini, lo studio delle condizioni di ottimalità. In termini molto generali, una condizione di ottimalità è un requisito, necessario e/o sufficiente, affinchè un punto x^* sia soluzione ottima, locale o globale, del problema. Logicamente, più vantaggiosa è la sua verifica, rispetto all'applicazione della definizione, più il requisito acquisisce significatività. Per questo motivo, solitamente le condizioni di ottimalità esprimono attraverso sistemi di equazioni o di disequazioni condizioni su matrici opportune. Studiare le condizioni di ottimo ha una duplice motivazione. Dal lato teorico, caratterizzando analiticamente le soluzioni ottime, è possibile svolgere, anche in assenza di soluzioni numeriche esplicite, analisi qualitative delle soluzioni ottime realizzando, ad esempio, un'analisi di sensitività [1]. Dal lato pratico, è possibile, grazie all'informazione

[1] Studio della variazione della soluzione ottima al variare dei coefficienti del problema

fornita da una condizione necessaria, restringere l'insieme delle soluzioni da valutare, costruendo così algoritmi finalizzati al soddisfacimento di tale condizione; oppure è possibile, mediante l'uso di una condizione sufficiente, mostrare l'ottimalità di una certa soluzione, fornendo così i criteri di arresto del processo risolutivo. Allo scopo di definire le condizioni di ottimalità è necessario introdurre alcuni concetti fondamentali. Per caratterizzare i minimi locali di una funzione, una definizione che ci torna molto utile è quella di direzione di discesa.

Definizione 12 *Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$ ed un punto $x \in \mathbb{R}^n$, un vettore $d \in \mathbb{R}^n$ si dice direzione di discesa per f in x se esiste $\bar{\lambda} > 0$ tale che*

$$f(x + \lambda d) < f(x), \quad \forall \lambda \in (0, \bar{\lambda}).$$

In sostanza, se ci troviamo in un punto x e ci spostiamo da x lungo d, la funzione, almeno per un tratto λ, decresce. Ne segue che se in un punto esistono direzioni di discesa, allora quel punto non può essere un punto di minimo locale. Nel nostro problema quanto detto si traduce nel fatto che, per migliorare il valore della funzione obiettivo, è necessario seguire le direzioni di discesa. A questo punto non ci resta che individuarle. Per spiegare la modalità con cui è possibile riconscerle, si introduce la seguente definizione

Definizione 13 (Derivata direzionale) *Dati una funzione $f : \mathbb{R}^n \to \mathbb{R}$, un vettore $d \in \mathbb{R}^n$ ed un punto x dove f è definita se esiste il limite*

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

allora tale limite prende il nome di derivata direzionale della funzione f nel punto x lungo la direzione d.

Per avere un'idea concreta di cosa rappresenti la derivata direzionale, andiamo a darne l'interpretazione geometrica in riferimento al nostro problema di ottimizzazione. Sia $x \in \mathcal{F}$, e sia $d \in \mathbb{R}^n$ una direzione. Consideriamo i punti che si incontrano spontandosi da x lungo d di un passo λ, ovvero i punti $x + \lambda d,\ \lambda > 0$. Il valore della derivata direzionale rappresenta il tasso di variazione della funzione obiettivo calcolato nel punto x, lungo la direzione d, ovvero la variazione di $f(x)$ allontanandosi da x lungo d. Si osservi che se $d = e_i$, cioè se la direzione coincide con l'i-esimo versore degli assi coordinati, allora la derivata direzionale coincide con la derivata parziale della funzione obiettivo nella componente i-esima. Rispetto alla derivata parziale, la derivata direzionale gode di un'importante caratteristica che la rende estremamente utile: può essere calcolata anche in punti in cui la funzione f non è differenziabile. Nel caso in cui invece sia possibile derivare la funzione, si ricava un'interessante relazione tra derivata direzionale e vettore gradiente $\nabla f(x)$.

Proprietà 14 (Gradiente di funzione derivabile) *Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$, derivabile con continuità in $\mathbb{R}^n$, la derivata direzionale di f lungo la direzione d è data da $\nabla f(x)^T d$.*

Sfruttando la proprietà appena esposta è possibile utilizzare le derivate prime per fornire una definizione alternativa di direzione di discesa per funzioni continuamente differenziabili.

Definizione 15 (Condizione di discesa) *Sia f una funzione continuamente differenziabile nell'intorno di un punto $x \in \mathbb{R}^n$ e sia $d \in \mathbb{R}^n$ un vettore non nullo. Se risulta*

$$\nabla f(x)^T d < 0 \tag{A.3}$$

allora d è una direzione di discesa per f in x.

Ricordando che $\nabla f(x)^T d = \|\nabla f(x)\| \, \|d\| \, cos\theta$, dove θ è l'angolo compreso tra $\nabla f(x)$ e d, da punto di vista geometrico la condizione (A.3) esprime il fatto che qualunque direzione formi un angolo ottuso con il gradiente della funzione obiettivo calcolato nel punto x è di discesa, e tutti i punti sufficientemente vicini ad x lungo d sono caratterizzati dall'avere un valore della funzione obiettivo minore di quello associato ad x. Quanto affermato equivale al fatto che se in un punto x il gradiente della funzione non è nullo, allora esiste almeno una direzione di discesa in x, e dunque il punto non è un minimo locale.

Teorema 16 (Condizioni necessarie di ottimalità del prim'ordine) *Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$, derivabile con continuità in $\mathbb{R}^n$, condizione necessaria affinchè il punto x^* sia un minimo locale per f è che risulti $\nabla f(x^*) = 0$.*

I punti che soddisfano le condizioni di ottimalità del prim'ordine vengono detti punti stazionari. In linea di principio però i punti che hanno gradiente nullo possono essere non solo di minimo locale, ma anche di massimo o di flesso. Per determinarne la natura e riuscire ad individuare solo quelli di minimo, è necessario sfruttare le informazioni fornite dalla matrice hessiana.

Teorema 17 (Condizioni necessarie di ottimalità del second'ordine) *Data una funzione $f : \mathbb{R}^n \to \mathbb{R}$ di classe $C^2(x^*)$, condizioni necessarie affinchè il punto x^* sia un minimo locale per f sono che risulti $\nabla f(x^*) = 0$ e che valga la relazione $d^T H(x^*) d > 0$ per ogni $d \in \mathbb{R}^n$.*

Si osservi che le condizioni appena enunciate, non sono sufficienti. Il fatto che un punto stazionario soddisfi la relazione $d^T H(x^*) > 0$ per ogni vettore di direzione d esclude soltanto che si tratti di un punto di massimo. Tuttavia se la funzione obiettivo è due volte differenziabile con continuità possiamo formulare le condizioni sufficienti di ottimalità del second'ordine.

Teorema 18 (Condizioni sufficienti di ottimalità del second'ordine) *Data una funzione $f : \mathbb{R}^n\mathbb{R}$ di classe $C^2(x^*)$, condizioni sufficienti (non necessarie) affinchè il punto x^* sia un minimo locale in senso stretto per f sono che risulti $\nabla f(x^*) = 0$ e che valga la relazione $d^T H(x^*) d > 0$ per ogni $d \in \mathbb{R}^n$.*

Quindi per scoprire se una data soluzione è ottima, basta controllare che il gradiente della funzione nel punto sia nullo e che la matrice H sia definita positiva. Siccome H è simmetrica, è possibile farlo verificando o che i determinanti delle

n sottomatrici quadrate che si ottengono considerando le matrici formate dalle sue prime i righe ed i colonne siano > 0 o che tutti gli autovalori siano > 0. La potenza dei problemi convessi consiste nel fatto che ogni minimo locale coincide con un minimo globale. Di conseguenza, avere buoni strumenti per riconoscere la convessità di un problema è basilare. Riportiamo in merito alcuni risultati.

Proprietà 19 (Funzione convessa) *Una funzione $f : \mathcal{F} \to \mathbb{R}$, derivabile con continuità nell'insieme convesso $\mathcal{F}$, è convessa su $\mathcal{F}$ se e solo se, comunque presi due punti $x, y \in \mathcal{F}$ vale*

$$f(y) - f(x) \geq \nabla f(x)^T (y - x).$$

Proprietà 20 (Funzione convessa) *Sia $\mathcal{F}$ un insieme convesso e sia $f : \mathcal{F} \to \mathbb{R}$ di classe $C^2(x)$, allora f è convessa se e solo se la matrice hessiana $H(x)$ è semidefinita positiva in x, per ogni punto $x \in \mathcal{F}$.*

A.4 Algoritmi di minimizzazione non vincolata

I problemi di ottimizzazione che si incontrano nella pratica sono di solito così complessi che risulta impossibile impiegare le condizioni di ottimalità appena esposte per calcolarne analiticamente la soluzione ottima, difatti la dimensione del problema [2]e l'eventuale presenza di funzioni non lineari sono generalmente tali da rendere impossibile il calcolo diretto. Per questo motivo, occore ricorrere ad algoritmi di tipo sequenziale, che terminano al soddisfacimento di un qualche criterio di convergenza empirico. In pratica, si sfruttano delle relazioni di tipo quantitativo che permettono di riconoscere, all'interno della successione dei punti visitati, quelli stazionari. Si osservi subito infatti che, in linea di principio, gli algoritmi di ottimizzazione non vincolata non forniscono alcuna garanzia che i punti trovati siano effettivamente di minimo, anche se molto spesso è possibile stabilire il soddisfacimento delle condizioni del seondo ordine. Lo schema di base degli algoritmi iterativi di discesa basati sulla ricerca lineare per problemi di ottimizzazione non vincolata è molto semplice

```
Procedure MetodoDiscesa (x_0)
  begin
   k=0
   while (∇f(x_k) ! = 0) do
     begin
      calcolo direzione discesa d_k ∈ R^n
      calcolo passo lungo la direzione discesa α_k ∈ R
      x_{k+1} = x_k + α_k d_k
      k++
     end
  end.
```

[2]Il numero di variabili e vincoli del problema

Gli aspetti critici, che differenziano i vari metodi di discesa, sono i criteri da usare per effettuare la scelta della direzione di discesa d_k e della lunghezza del passo α_k. Peraltro, da queste scelte dipendono le caratteristiche fondamentali dell'algoritmo, la convergenza a punti stazionari della successione dei punti x_k e la rapidità di convergenza.

A.5 Convergenza e rapidità di convergenza

Nell'analisi delle prestazioni di un algoritmo di discesa, un importante criterio di classificazione è stabilire se esso converga. In particolare, nella nostra trattazione e più in generale nell'ambito della programmazione non lineare, se esso converga ad un minimo locale. Una caratterizzazione della convergenza si ha in relazione alla scelta del punto iniziale della successione x_k.

Definizione 21 (Algoritmo globalmente convergente) *Un algoritmo è globalmente convergente se esso converge per qualunque $x_0 \in \mathbb{R}^n$.*

Definizione 22 (Algoritmo localmente convergente) *Un algoritmo è localmente convergente se esso converge per solo per $x_0 \in I(x^*)$, dove $I(x^*)$ è un opportuno intorno di un punto $x^* \in PS$.*

Nel valutare le prestazioni di un algoritmo, oltre alla convergenza, occorre poi prendere in considerazione la sua velocità di convergenza, ovvero di caratterizzare la rapidità con cui, posto che le condizioni di convergenza siano soddisfatte, l'algoritmo converga a partire dal punto inziale scelto. Infatti dati due algoritmi che risolvono lo stesso problema, il numero di iterazioni richiesto può essere molto diverso. Nell'ambito della programmazione lineare, lineare intera e per problemi di ottimizzazione su grafo, in cui gli algoritmi impiegano al più un numero esponenziale, ma finito, di iterazioni, è possibile fornire la definizione di efficienza di un algoritmo sulla base del numero di iterazioni necessarie per ottenere, al caso pessimo, la soluzione ottima. Nel caso della programmazione non lineare, fatta eccezione per alcuni casi particolare quali ad esempio il metodo di Newton per funzioni quadratiche, gli algoritmi possono produrre una sequenza infinita di punti, con conseguente infinito numero di iterazioni, e dunque la definizione di efficienza deve essere necessariamente rivista. I metodi più utilizzati per misurare la velocità di convergenza della successione x_k al punto limite x^*, a cui l'algoritmo dovrebbe convergere, si basano sull'analisi di come si riduce il termine $\|x_{k+1} - x^*\|$ rispetto al termine $\|x_k - x^*\|$, ovvero sull'analisi del rapporto, ad un'iterazione e alla successiva, tra gli scostamenti esistenti tra la soluzione corrente ed il punto limite, cioè

$$\frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|}$$

A.6 Derminazione del passo lungo la direzione di discesa

Affrontiamo, in prima istanza, il problema di determinare il passo α_k, supponendo di conoscere la direzione di discesa d_k. Dal momento che la ricerca di α_k avviene lungo una linea, questa ricerca prende il nome di line search. Gli algoritmi di line search consistono nel testare iterativamente diversi valori di α, sino a che alcuni criteri di arresto non risultano soddisfatti. Fissati quindi x_k e d_k, la scelta del passo è fatta sulla base dell'andamento di f lungo la direzione di ricerca. Esprimiamo quindi la variazione della funzione obiettivo al variare di α

$$\Phi(\alpha) = f(x_k + \alpha d_k).$$

Poich in un determinato punto x e per una determinata direzione di dicesa d solo il valore di α può variare con continuità, ci si può ricondurre alla risoluzione di un problema di programmazione non lineare nella sola variabile α

$$\min \Phi(\alpha) = f(x_k + \alpha d_k) \quad \alpha > 0$$

Nell'ipotesi che a funzione obiettivo sia differenziabile, esiste la derivata di $\Phi(\alpha)$ rispetto ad α, indicata con $\Phi'(\alpha)$, che può essere posta nella forma

$$\Phi'(\alpha) = \nabla f(x_k + \alpha d_k)^T d_k$$

e rappresenta il tasso di variazione del valore della funzione obiettivo mano a mano che ci si allontana da x_k lungo d_k. In particolare, per $\alpha = 0$, coincide con la derivata direzionale di f calcolata nel punto x_k. In generale, siccome la determinazione del minimo,locale o globale, della funzione $\Phi(\alpha)$ per valori di $\alpha > 0$ è molto costoso, si impiegano strategie che individuano un intervallo di valori accettabili di α ai fini dell'ottenimento della convergenza per α_k. . Riportiamo velocemente le condizioni che un buon algoritmo deve determinare α deve possedere. Per impedire che la successione di valori $\Phi(\alpha_x)$ converga al valore sbagliato, è necessario che la riduzione della funzione sia proporzionale sia alla lunghezza del passo, che alla derivata direzionale della funzione calcolata nel punto da cui ci si sta allontanando.

Condizione 23 (Condizione sufficiente di riduzione della funzione) *Sia f la funzione di cui si sta cercando il minimo. Allora ad ogni iterazione dell'algoritmo, il punto incrementato deve soddisfare la relazione*

$$f(x + \alpha d) \leq f(x) + c_1 \nabla f(x)^T d, \quad c_1 \in (0, 1). \tag{A.4}$$

La disuguaglianza (A.4), detta condizione di Armijo, tuttavia non basta da sola ad assicurare che l'algoritmo faccia progressi ragionevoli. Per evitare che l'algoritmo effettui passi troppo piccoli, è necessario imporre anche una seconda condizione

Condizione 24 (Condizione di curvatura) *Ad ogni iterazione dell'algoritmo, il punto incrementato deve soddisare la seguente disuguaglianza*

$$\triangledown f(x + \alpha d)^T d \geq c_2 \triangledown f(x)^T d, \quad c_2 \in (c_1, 1).$$

In sostanza, questa condizione vincola il passo ad essere abbastanza lungo da percepire una significativa diminuzione, in valore assoluto, della derivata direzionale, ovvero un avvicinamento al minimo della funzione obiettivo. Le due condizioni appena esposte, prendono il nome di condizioni di Wolfe. La scelta del valore con cui inizializzare α_0 dipende dalla tecnica impiegata per scegliere la direzione di discesa.

A.7 Scelta della direzione di discesa

A.7.1 Il metodo del gradiente

Il metodo del gradiente è uno fra i primi metodi proposti per la minimizzazione non vincolata e si basa sull'uso della direzione di ricerca

$$d_k = -\triangledown f(x_k)$$

ossia dell'antigradiente di f nel punto x_k. Questo metodo prende anche il nome di metodo della discesa più rapida: se infatti osserviamo la direzione dell'antigradiente, possiamo accorgerci che, tra tutte le direzioni che hanno norma euclidea unitaria, è quella che minimizza la la derivata direzionale di x_k. L'interesse per l'antigradiente risiede nel fatto che, nell'ipotesi che il gradiente sia continuo, esso costituisce una direzione di discesa continua rispetto ad x che si annulla se e solo se x è un punto stazionario. Il metodo del gradiente può essere schematizzato come segue

Procedure MetodoGradiente (x_0)
begin
k=0
while ($\nabla f(x_k)$! = 0) **do**
begin
$d_k = -\nabla f(x_k)$
calcolo passo lungo la direzione discesa $\alpha_k \in \mathbb{R}$
$x_{k+1} = x_k + \alpha_k d_k$
$k++$
end
end.

Siccome così facendo si ha che, ad ogni iterazione, si segue la direzione che massimizza il decremento della funzione obiettivo, questa scelta sembra, nel contesto dei metodi di discesa, la più ovvia. Tuttavia non sempre è la più vincente. In prima analisi, si osservi intanto che scegliere un passo che garantisca una semplice riduzione della funzione obiettivo tra un'iterazione e l'altra, non è sufficiente alla convergenza del metodo.

Appendix B

Ottimizzazione vincolata

B.1 Introduzione

Si prenda ora in considerazione il generico problema di ottimizzazione vincolata

$$\min f(x) : \ x \in S$$

con $S \subset \mathbb{R}^n$. Procediamo con lo studiare le condizioni di ottimalità. Come visto sinora, nel caso di problemi non vincolati tutti i minimi locali soddisfano le condizioni necessarie di ottimalità e, almeno il linea di principio, possono essere ricercati tra i punti stazionari del problema. Nel caso vincolato, invece, non è sempre possibile ricavare tutti i minimi locali anche nel caso in cui risulta possibile imporre il soddisfacimento delle condizioni analitiche. Nel seguito, illustriamo in primo luogo le condizioni di ottimalità in relazione a problemi in cui l'insieme ammissibile S è un generico insieme convesso, per poi passare ad analizzare separatamente il caso in cui la regione ammissibile sia descritta da soli vincoli di disuguaglianza, da soli vincoli di uguaglianza ed infine da entrambi.

B.2 Le condizioni di Karush-Kuhn-Tucker

Una classe significativa di problemi di programmazione matematica è quella in cui l'insieme ammissibile è un poliedro. Si consideri un poliedro definito come

$$S = x \in \mathbb{R}^n \quad : \quad Ax \geq b$$

dove A è la matrice reale $m \times n$ e $b \in \mathbb{R}^m$. Il problema in esame è quindi

$$\min f(x) : \ Ax \geq b \tag{B.1}$$

In cui f è una funzione continuamente differenziabile. Utilizzando la caratterizzazione di direzioni ammissibili per un poliedro è possibile ricavare la condizione necessaria di minimo globale su un poliedro

Teorema 25 (Condizione necessaria e sufficiente di minimo locale su un poliedro) *Sia f una funzione convessa. Un punto x^* è minimo locale se e solo se non esiste una soluzione $d \in \mathbb{R}^n$ al sistema di disequazioni lineare*

$$\begin{array}{c} A_{I(x^*)}d \geq 0 \\ \nabla f(x^*)^T d < 0 \end{array} \tag{B.2}$$

dove $A_{I(x^*)}$ è la matrice dei vincoli attivi $A_{I(x*)} = (a_i^T)_{iI(x^*)}$ di dimensione $|I(x^*)| \times n$ e $I(x^*) = i :\ a_i^T x^* = b_i$ è l'insieme degli indici di tutti i vincoli attivi in x^*.

Il teorema appena enunciato descrive le condizioni di ottimo per problemi con vincoli lineari formulandole come la non esistenza di soluzione di un sistema di disequazioni lineari. In realtà queste condizioni possono essere riscritte, tramite i cosiddetti teoremi dell'alternativa, come condizioni di esistenza di soluzioni di un diverso sistema di disequazioni lineari. Questi teoremi consentono di ridurre il problema della non esistenza di soluzioni di un sistema di equazioni e disequazioni lineari a quello dell'esistenza di soluzioni di un altro sistema lineare. Tra i teoremi dell'alternativa per sistemi di disequazioni lineari, uno tra i più noti, e anche quello che verrà impiegato nel corso della trattazione, è il Lemma di Farkas che può essere enunciato come segue

Teorema 26 (Lemma di Farkas) *Sia B una matrice $p \times n$ e $g \in \mathbb{R}^n$. Il sistema*

$$Bd \geq 0 \quad g^T d < 0 \tag{B.3}$$

non ha soluzione $d \in \mathbb{R}^n$ se e solo se il sistema

$$B^T u = g \quad u \geq 0 \tag{B.4}$$

ha soluzione $u \in \mathbb{R}^p$.

Sfruttando il lemma appena esposto, facendo coincidere il sistema (B.2) con il sistema (B.3), è possibile affermare che non esiste una soluzione $d \in \mathbb{R}^n$ al sistema di disequazioni lineari (B.2) se e solo se esiste una soluzione al sistema

$$\begin{array}{c} A^T_{I(x*)}u = \nabla f(x^*) \\ u \geq 0 \end{array} \tag{B.5}$$

dove u è un vettore di dimensione $|I(x^*)|$. Utilizzando il Lemma di Farkas, risulta possibile introdurre le condizioni di ottimo per il problema di ottimizzazione vincolata descritto da soli vincoli di disuguaglianza, note come condizioni di Karush-Kuhn-Tucker (KKT)

Teorema 27 (Condizioni di Karush-Kuhn-Tucker per (B.1)) *Sia x^* un punto di minimo locale per il problema (B.1). Allora esiste un vettore $s^* \in \mathbb{R}^m$ tale che risultino soddisfatte le seguenti condizioni:*

1. *$Ax^* \geq b$*

2. $\nabla f(x^*) - A^T s^* = 0$

3. $s^* \geq 0$

4. $s_i^*(b_i - a_i^T x^*) = 0 \quad i = 1, , m$

Si osservi che la condizione (ii) delle condizioni 27 esprime il fatto che in un punto di minimo locale il gradiente della funzione è esprimibile come combinazione non negativa conica dei gradienti dei vincoli attivi in x^*

$$\nabla f(x^*) = \sum_{i \in I(x^*)} s_i^* a_i \quad s_i^* \geq 0$$

ovvero, all'ottimo il gradiente della funzione è contenuto nel cono individuato dai gradienti dei vincoli attivi in x^*. La (iv) è nota come condizione di complementarità ed equivale al fatto che s_i può essere positivo solo se il corrispondente vincolo è attivo e deve annullarsi in caso contrario. Tenendo conto delle condizioni (i) e (iii) è possibile riscrivere la (iv) come

$$s^{*T}(b - Ax^*) = \sum_{i=1}^{m} s_i^*(b_i - a_i^T x^*) = 0$$

Le condizioni KKT si possono esprimere in maniera equivalente mediante la nozione di funzione Lagrangiana.

Definizione 28 (Funzione Lagrangiana per (B.1)) *La funzione Lagrangiana per il problema (B.1) è definita come*

$$L(x, s) = f(x) + s^T(b - Ax)$$

Denotando con $\nabla_x L(x, s) = \nabla f(x) - A^T s$ il gradiente della funzione lagrangiana rispetto ad x, la condizione (ii) esprime il fatto che nel punto (x^*, s^*), la quantità $\nabla_x L(x, s)$ deve annullarsi infatti

$$\nabla f(x^*) - A^T s^* = 0 \quad \Leftrightarrow \quad \nabla f(x^*) - (-\nabla_x L(x, s) + \nabla f(x^*)) = 0 \quad \Leftrightarrow \quad \nabla_x L(x, s) = 0.$$

Ricordando che un punto stazionario di una funzione $\varphi : \mathbb{R}^n \to \mathbb{R}$, continuamente differenziabile sul codominio, è un punto in cui il gradiente della funzione φ si annulla, è possibile osservare che la coppia (x^*, s^*) costituisce un punto stazionario della funzione Lagrangiana. Di conseguenza, in punto x^* che soddisfa le condizioni KKT si ha

$$L(x^*, s^*) = f(x^*) + s^*(b - Ax^*) = f(x^*)$$

Siccome i punti candidati ad essere punti di minimo di un problema (B.1) possono essere determinati risolvendo le condizioni KKT, si introduce la seguente definizione

Definizione 29 (Punto di Karush-Kuhn-Tucker) *Un punto x' è detto punto di KKT del problema (B.1) se esiste $s' \in \mathbb{R}^m$ tale che valgano le seguenti condizioni:*

1. $Ax' \geq b$
2. $\nabla L(x', s') = 0$
3. $s' \geq 0$
4. $s'(b - Ax') = 0$.

Giacch il teorema 25 diviene, nell'ipotesi di f convessa, condizione necessaria e sufficiente di ottimo globale è possibile affermare che le condizioni KKT, sempre nell'ipotesi di f convessa, diventano necessarie e sufficienti di ottimo globale. Se inoltre la f è strettamente convessa allora il punto x^* che soddisfa le KKT è l'unico punto di minimo globale della funzione.
Dalle condizioni KKT per problemi nella forma (B.1) è possibile derivare le condizioni KKT per un problema con un poliedro più generale. Si consideri il caso di un poliedro descritto da soli vincoli di uguaglianza

$$\min f(x) \ : \ Ax = b \tag{B.6}$$

dove A è una matrice di dimensione $m \times n$. È noto che il set di vincoli di uguaglianza può essere equivalentemente riscritto come $Ax \geq b$ e $-Ax \geq -b$. Sfruttando questa nozione, si riscrive (B.6) come

$$\min f(x) \ : \ Ax \geq b, \ \ -Ax \geq -b$$

associando ai vincoli i moltiplicatori $y_1, y_2 \in \mathbb{R}^m$ con $y_1, y_2 \geq 0$ la funzione Lagrangiana per (B.6) è

$$L(x, y_1, y_2) = f(x) + y_1^T(b - Ax) + y_2^T(-b + Ax) = f(x) + (y_1 - y_2)^T(b - Ax)$$

operando la seguente sostituzione $y = y_1 - y_2$, si ottiene

$$L(x, y) = f(x) + y^T(b - Ax).$$

Considerato che y è definito come differenza di due quantità non negative, è del tutto ovvio che le entrate di tale vettore possono assumere un valore qualunque (positivo, negativo, nullo). è inoltre banalmente dimostrabile che la condizione di complementarità è soddisfatta in ogni punto ammissibile.

Teorema 30 (Condizioni necessarie di Lagrange) *Sia x^* un punto di minimo locale del problema (B.6). Allora esiste un vettore $y^* \in \mathbb{R}^m$ tale che*

$$\nabla_x L(x^*, y^*) = \nabla f(x^*) - A^T y^* = 0$$

Tramite la caratterizzazione dell'ottimo per i problemi (B.1) e (B.6) arriviamo così ad enunciare le condizioni KKT per un problema in forma standard

$$\min f(x): \quad Ax = b, \quad x \geq 0 \tag{B.7}$$

la cui funzione Lagrangiana è definita come

$$L(x, y, s) = f(x) + y^T(Ax - b) - s^T x$$

Teorema 31 (Condizioni di Karush-Kuhn-Tucker per (B.7)) *Sia x^* un punto di minimo locale per il problema (B.7). Allora esistono un vettore $y^* \in \mathbb{R}^n$ e $s^* \in \mathbb{R}^m$ tale che risultino soddisfatte le seguenti condizioni:*

1. $Ax^* = b, \quad x^* \geq 0$ *ammissibilità*
2. $\nabla_x L(x^*, y^*, s^*) = f(x^*) + A^T y^* - s^* = 0$ *stazionarietà*
3. $s^* \geq 0$
4. $s^{*T} x^* = 0$ *complementarità*

Bibliography

[1] C. T. Kelley, Iterative Methods for Linear and Nonlinear Equations,vol. 16 of Frontiers in Applied Mathematics, SIAM, Philadelphia, 1995.

[2] C. Rolli, Il precondizionamento di un sistema lineare

[3] R. Ahuja, T.L. Magnanti, J.B. Orlin, M.R. Reddy, Applications of Network Optimization, OR 300-94, 1994.

[4] Ke Chen, Matrix Preconditioning Techniques and Applications, Cambridge University Press, 2005.

[5] N. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica 4(4) 373-395, 1984.

[6] F. A. Potra, S. J. Wright, Interior-Point methods, Journal of Computational and Applied Mathematics 124(1-2) 281-302, 2000.

[7] S. J. Wright, Primal-Dual Interior-Point Methods, SIAM, Philadelphia, 1997.

[8] A. Frangioni, C. Gentile, Interior Point Methods for Network Problems, Technical Report 539, IASI, CNR, Roma, 2000.

[9] A. Frangioni, C. Gentile, Prim-based support-graph preconditioners for min-cost flow problems, Technical Report 627, IASI, CNR, Roma, 2005

[10] `http://www.di.unipi.it/optimize/Data/MCF.html`

[11] `http://lpsolve.sourceforge.net/5.5/DIMACS_maxf.htm`

[12] `http://www.di.unipi.it/optimize/Software/MCF.html#MCFSimplex`

[13] Jordi Castro, Universitat Politècnica de Catalunya. Consulenza privata.