Note del corso di Ricerca Operativa

Laurea Magistrale in Matematica
Università degli Studi di Padova

Marco Di Summa

A.A. 2011-2012

Indice

Capitolo 1

Programmazione lineare

1.1 Richiami di teoria

In questa sezione verranno richiamati i concetti base e i teoremi principali della teoria della Programmazione Lineare. La presentazione di questi argomenti è limitata agli aspetti che troveranno applicazione nella parte successiva del corso. I risultati vengono forniti senza dimostrazione. Per approfondimenti si può fare riferimento alle dispense del corso di Ottimizzazione Discreta.

1.1.1 Definizioni e generalità

Un *programma lineare* è un problema di ottimizzazione vincolata in cui la funzione da massimizzare (o minimizzare) è lineare e i vincoli sono descritti da equazioni e/o disequazioni lineari:

$$\begin{aligned} \max \text{ (o min)} \quad & c_1 x_1 + \cdots + c_n x_n && (1.1)\\ \text{soggetto a} \quad & a_{11} x_1 + \cdots + a_{1n} x_n \sim b_1 && (1.2)\\ & \vdots \qquad\qquad \vdots \\ & a_{m1} x_1 + \cdots + a_{mn} x_n \sim b_m, && (1.3) \end{aligned}$$

dove ciascuno dei simboli "$\sim$" sta per uno degli operatori "$\geq$", "$\leq$" o "$=$" (si noti che non sono ammesse disequazioni strette). Nel problema (1.1)–(1.3), $x_1, \ldots, x_n$ sono le *variabili*, (1.1) è la *funzione obiettivo* e (1.2)–(1.3) sono i *vincoli*. Tutti i dati a_{ij}, b_i, c_j ($i = 1, \ldots, m$, $j = 1, \ldots, n$) sono numeri reali; le variabili $x_1, \ldots, x_n$ possono prendere qualunque valore reale, purché esse soddisfino tutti i vincoli. Si noti che l'espressione (1.1) può essere scritta più concisamente come $c^\top x$.[1]

Ogni vettore $x \in \mathbb{R}^n$ che soddisfi (1.2)–(1.3) è detto una *soluzione ammissibile* del problema. L'insieme di tutte le soluzioni ammissibili di un programma lineare è detto *regione ammissibile*. Un vettore $\bar{x} \in \mathbb{R}^n$ è una soluzione ottima del programma lineare se $\bar{x}$ è ammissibile e $c^\top \bar{x} \geq c^\top x$ per ogni x ammissibile (la condizione è $c^\top \bar{x} \leq c^\top x$ se il problema è di minimizzazione). Il corrispondente valore $c^\top \bar{x}$ è detto *valore ottimo*.

[1] In queste note tutti i vettori sono vettori-colonnna.

Teorema 1.1 *Dato un programma lineare, vale una e una sola delle seguenti alternative:*

(a) il problema ha almeno una soluzione ottima;

(b) il problema è inammissibile*, cioè non ha soluzioni ammissibili;*

(c) il problema è illimitato*, cioè per ogni $K \in \mathbb{R}$ esiste una soluzione ammissibile x tale che $c^\top x > K$ ($c^\top x < K$ se il problema è di minimizzazione).*

1.1.2 Forma standard, soluzioni di base e metodo del simplesso

Nel seguito, dati una matrice $A \in \mathbb{R}^{m\times n}$ ed un insieme di indici $I \subseteq \{1,\dots,n\}$, la notazione A_I indicherà la sottomatrice di A formata dalle colonne con indice in I. Analogamente, dato un vettore $x \in \mathbb{R}^n$, x_I indicherà il sottovettore formato dalle sole componenti x_i tali che $i \in I$.

Ogni programma lineare è equivalente ad un programma lineare in *forma standard*, cioè del tipo

$$\max\ c^\top x \tag{1.4}$$
$$\text{s.a}\ Ax = b \tag{1.5}$$
$$x \geq \mathbf{0}, \tag{1.6}$$

dove $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ e x è un vettore di variabili in $\mathbb{R}^n$. Nozioni di base di algebra lineare implicano che se il problema è ammissibile allora si può assumere senza perdita di generalità che le righe di A siano linearmente indipendenti (da cui, in particolare, $m \leq n$).

Dato un programma lineare in forma standard (1.4)–(1.6) in cui le righe di A sono linearmente indipendenti, si definisce *base* un qualunque sottoinsieme di indici $B \subseteq \{1,\dots,n\}$ tale che $|B| = m$ e la matrice quadrata A_B è invertibile. (Si noti che almeno una base esiste, perché il rango di A è m.) La corrispondente *non-base* è $N = \{1,\dots,n\} \setminus B$. Si dice anche che la sottomatrice A_B è una base della matrice A.

Data una base B, vale la catena di equivalenze

$$Ax = b \iff A_B x_B + A_N x_N = b \iff x_B = A_B^{-1} b - A_B^{-1} A_N x_N. \tag{1.7}$$

Si definisce *soluzione di base* associata alla base B l'unica soluzione $\bar{x}$ del sistema $Ax = b$ ottenuta ponendo $\bar{x}_N = \mathbf{0}$. L'unicità segue da (1.7), che mostra anche che deve valere $\bar{x}_B = A_B^{-1} b$. Si noti che una soluzione di base non è necessariamente ammissibile, in quanto i vincoli (1.6) potrebbero essere violati. La soluzione di base $\bar{x}$ è dunque ammissibile se e solo se $\bar{x}_B = A_B^{-1} b \geq \mathbf{0}$.

Teorema 1.2 *Se un programma lineare in forma standard ha una soluzione ottima, allora c'è almeno una soluzione ottima che è una soluzione di base (ovviamente ammissibile).*

Il teorema appena enunciato permette di restringere la ricerca dell'ottimo alle sole soluzioni di base.

L'algoritmo più usato per risolvere problemi di Programmazione Lineare è il *metodo del simplesso*. Senza addentrarsi nei dettagli, si daranno qui solo poche informazioni essenziali su questo algoritmo.

- Il metodo del simplesso parte da una qualunque soluzione di base ammissibile (una tale soluzione di partenza viene trovata con la cosiddetta "fase uno" del metodo del simplesso; se non ci sono soluzioni ammissibili, la fase uno fornisce una prova di questo fatto).
- Ad ogni iterazione, il metodo del simplesso si muove dalla soluzione di base ammissibile corrente ad una nuova soluzione di base ammissibile, ottenuta sostituendo un unico indice della base corrente con un nuovo indice.
- Ad ogni iterazione, la nuova soluzione di base ammissibile ottenuta ha un valore $c^\top x$ non peggiore di quello della soluzione precedente.
- Dopo un numero finito di iterazioni, il metodo del simplesso restituisce una soluzione di base ottima (oppure fornisce una prova dell'illimitatezza del problema).

In questo corso verrà usato più volte il fatto che la soluzione ottima restituita dal metodo del simplesso è una soluzione di base. La corrispondente base (che potrebbe non essere unica) è detta una *base ottima*.

1.1.3 Dualità

Data una matrice $A \in \mathbb{R}^{m\times n}$ con componenti a_{ij}, indichiamo con $a_i^\top$ la i-esima riga di A e con A_j la j-esima colonna di A.

Si consideri un programma lineare nella sua forma più generale, con n variabili ed m vincoli (assumiamo soltanto, per il momento, che il problema sia di massimizzazione):

$$\begin{aligned}
\max \quad & c^\top x && && (1.8)\\
\text{s.a} \quad & a_i^\top x \le b_i, && i = 1,\dots,m_1 && (1.9)\\
& a_i^\top x \ge b_i, && i = m_1+1,\dots,m_2 && (1.10)\\
& a_i^\top x = b_i, && i = m_2+1,\dots,m && (1.11)\\
& x_j \ge 0, && j = 1,\dots,n_1 && (1.12)\\
& x_j \le 0, && j = n_1+1,\dots,n_2 && (1.13)\\
& x_j \text{ libera}, && j = n_2+1,\dots,n. && (1.14)
\end{aligned}$$

Si noti che questo problema è esattamente il programma lineare (1.1)–(1.3), ma i simboli "$\sim$" sono stati resi espliciti ed eventuali vincoli sul segno delle variabili sono stati scritti separatamente. Quest'ultima scelta non è strettamente necessaria, ma rende la scrittura del duale più agevole per la maggior parte dei programmi lineari.

Il *duale* di (1.8)–(1.14) (che per contrasto viene chiamato *primale*) è un altro programma lineare, costruito seguendo le regole qui sotto elencate:

- Il duale di un problema di massimo è un problema di minimo; viceversa, il duale di un problema di minimo è un problema di massimo.
- Ad ogni vincolo (1.9)–(1.11) del primale è associata una variabile u_i del duale: il tipo di vincolo primale determina il segno della corrispondente variabile duale; viceversa, ad ogni variabile x_j del primale è associato un vincolo del duale: il segno della variabile primale determina il tipo di vincolo duale.
- Il vettore c che definisce la funzione obiettivo del primale diventa il vettore dei termini noti del duale; viceversa, il vettore b dei termini noti del primale diventa il vettore della funzione obiettivo del duale.
- Se i vincoli (1.9)–(1.11) sono definiti da una matrice $A \in \mathbb{R}^{m \times n}$, quelli del duale sono definiti dalla matrice trasposta $A^\top$.

La tabella seguente fornisce le regole complete per costruire il duale di un programma lineare:

max $c^\top x$	min $b^\top u$	
$a_i^\top x \leq b_i$	$u_i \geq 0$	$i = 1, \ldots, m_1$
$a_i^\top x \geq b_i$	$u_i \leq 0$	$i = m_1 + 1, \ldots, m_2$
$a_i^\top x = b_i$	u_i libera	$i = m_2 + 1, \ldots, m$
$x_j \geq 0$	$A_j^\top u \geq c_j$	$j = 1, \ldots, n_1$
$x_j \leq 0$	$A_j^\top u \leq c_j$	$j = n_1 + 1, \ldots, n_2$
x_j libera	$A_j^\top u = c_j$	$j = n_2 + 1, \ldots, n$

Le regole di costruzione del duale di un problema di minimo si ottengono leggendo la tabella da destra a sinistra. Inoltre, è facile verificare che il duale del duale coincide con il primale.

Vengono dati qui sotto i principali teoremi sulla teoria della dualità in Programmazione Lineare. In tutti gli enunciati che seguono, si fa riferimento ad una coppia di programmi lineari primale-duale come nella tabella sopra riportata.

Teorema 1.3 (Teorema di Dualità Debole) *Se x è ammissibile primale e u è ammissibile duale, allora $c^\top x \leq b^\top u$.*

Corollario 1.4 *Se x è ammissibile primale, u è ammissibile duale e $c^\top x = b^\top u$, allora x e u sono soluzioni ottime dei rispettivi problemi.*

Teorema 1.5 (Teorema di Dualità Forte) *Il primale ha una soluzione ottima se e solo se anche il duale ce l'ha, e in tal caso i valori ottimi dei due problemi coincidono.*

Grazie ai teoremi enunciati sopra, possiamo compilare la seguente tabella, che indica quali combinazioni sono possibili (marcate col simbolo "✓") e quali no (marcate con "×") per un problema primale ed il suo duale:

		Primale		
		Sol. ottima	Inammissibile	Illimitato
Duale	Sol. ottima	✓	×	×
	Inammissibile	×	✓	✓
	Illimitato	×	✓	×

Corollario 1.6 (Teorema degli Scarti Complementari) *Siano x e u soluzioni ammissibili per il primale ed il duale rispettivamente. Allora x e u sono ottime per i rispettivi problemi se e solo se soddisfano gli* scarti complementari*:*

- *per ogni $j = 1, \ldots, n$, se $x_j \neq 0$ allora $A_j^\top u = c_j$;*
- *per ogni $i = 1, \ldots, m$, se $u_i \neq 0$ allora $a_i^\top x = b_i$.*

A parole, le condizioni sugli scarti complementari richiedono che se una variabile primale (risp., duale) è diversa da zero allora il corrispondente vincolo duale (risp., primale) è soddisfatto ad uguaglianza.

Il caso della forma standard

Si ricordi che un programma lineare in forma standard è del tipo

$$\begin{aligned} \max\ & c^\top x \\ \text{s.a}\ & Ax = b \\ & x \geq \mathbf{0}. \end{aligned}$$

In tal caso, seguendo le regole date sopra, il duale assume la forma seguente:

$$\begin{aligned} \min\ & b^\top u \\ \text{s.a}\ & A^\top u \geq c. \end{aligned}$$

Come detto, il metodo del simplesso restituisce una soluzione ottima (se ne esiste una) che è una soluzione di base associata ad una certa base B. La soluzione ottima sarà dunque un vettore $\bar{x}$ definito da $\bar{x}_B = A_B^{-1} b$, $\bar{x}_N = \mathbf{0}$. È possibile dimostrare che allora la soluzione duale $\bar{u}$ definita da

$$\bar{u} = A_B^{-\top} c_B$$

è una soluzione ottima per il problema duale.

1.2 Interpretazione economica della dualità

1.2.1 Il problema duale come mercato ombra

Quando un programma lineare nasce come modello che descrive un problema di natura economica, il duale ammette generalmente un'interpretazione economica. Tale interpretazione dipende dal problema specifico (alcuni esempi

saranno dati nella Sezione 1.4), ma di norma è possibile vedere il duale come il programma lineare che descrive il problema di un operatore intenzionato ad entrare nel mercato, il quale ci propone un'alternativa al nostro business (ci si riferisce a questa situazione come *mercato ombra*). I vincoli del duale possono essere letti come le condizioni che assicurano la concorrenzialità della proposta (cioè il fatto che l'offerta sia economicamente interessante per noi), mentre la funzione obiettivo del duale descriverà ovviamente il tentativo dell'operatore di trarre il massimo profitto dalla sua entrata nel mercato. Questa interpretazione diventerà comunque più chiara grazie agli esempi della Sezione 1.4.

1.2.2 Variabili duali come prezzi di equilibrio

Le variabili duali hanno anche un'altra interessante interpretazione. Diamo qui l'enunciato per il caso della forma standard, ma il risultato vale in generale, dato che un programma lineare può sempre essere messo in forma standard.

Proposizione 1.7 *Si considerino un programma lineare in forma standard ed il suo duale:*

$$\begin{array}{ll} \max\ c^\top x & \min\ \ b^\top u \\ s.a\ Ax = b & s.a\ A^\top u \geq c. \\ \quad x \geq \mathbf{0} & \end{array}$$

Sia B una base ottima del problema primale e siano $\bar{x}, \bar{u}$ le corrispondenti soluzioni ottime primale e duale. Allora ciascuna variabile duale $\bar{u}_i$ indica di quanto varierebbe il valore ottimo se il termine noto del corrispondente vincolo primale b_i aumentasse di una unità e la base ottima restasse la stessa.

Dimostrazione. Si ricordi che $\bar{x}_B = A_B^{-1}b$, $\bar{x}_N = \mathbf{0}$ e $\bar{u} = A_B^{-\top}c_B$. Sia inoltre z il valore ottimo del problema, cioè

$$z = c^\top \bar{x} = c_B^\top \bar{x}_B + c_N^\top \bar{x}_N = c_B^\top A_B^{-1} b.$$

Se il termine noto b_i viene aumentato di una unità, otteniamo un nuovo vettore dei termini noti $b' = b + e_i$. Assumendo che la base B sia ottima anche per il problema perturbato, la nuova soluzione ottima primale sarà $\bar{x}'_B = A_B^{-1}b'$, $\bar{x}'_N = \mathbf{0}$. Allora il nuovo valore ottimo z' sarà

$$z' = c^\top \bar{x}' = c_B^\top \bar{x}'_B + c_N^\top \bar{x}'_N = c_B^\top A_B^{-1} b' = c_B^\top A_B^{-1} b + c_B^\top A_B^{-1} e_i = z + \bar{u}_i,$$

dove l'ultima uguaglianza vale perché $c_B^\top A_B^{-1} e_i = e_i^\top A_B^{-\top} c_B = e_i^\top \bar{u} = \bar{u}_i$. Dunque la variazione del valore ottimo è pari a $z' - z = \bar{u}_i$. □

Si tenga ben presente che tale interpretazione delle variabili duali è valida solo se la base ottima rimane la stessa quando il termine noto del primale viene perturbato. Vedremo nella Sezione 1.3.1 come stabilire per quali perturbazioni del termine noto la base ottima rimane effettivamente invariata.

Alla luce della Proposizione 1.7, le variabili duali possono essere interpretate come *prezzi di equilibrio*: il valore di $\bar{u}_i$ ci dice quanto saremmo disposti a pagare

per incrementare di una unità il termine noto b_i (ammesso che la base ottima resti la stessa): infatti, poiché il valore ottimo aumenterebbe esattamente di $\bar{u}_i$, non saremmo certo disposti a pagare più di $\bar{u}_i$ per incrementare b_i. Anche questo concetto verrà comunque esemplificato nella Sezione 1.4.

1.3 Analisi della sensitività

Sebbene sia considerato un algoritmo abbastanza efficace, il metodo del simplesso può richiedere numerosissime iterazioni quando il programma lineare in esame è di notevoli dimensioni. A volte, dopo aver risolto un programma lineare, ci si accorge che uno o più dati del problema sono cambiati. Se il programma lineare è molto grosso, pensare di risolverlo nuovamente ogni volta che i dati subiscono modifiche può essere proibitivo. L'*analisi della sensitività* ha lo scopo di fornire informazioni su quanto è possibile perturbare i dati del problema senza che ciò ci costringa a risolverlo nuovamente.

Consideriamo un problema di programmazione lineare in forma standard ed il suo duale:

$$\begin{array}{ll} \max\ c^\top x & \min\ \ b^\top u \\ \text{s.a } Ax = b & \text{s.a } A^\top u \geq c. \\ \quad x \geq \mathbf{0} & \end{array}$$

Sia B una base ottima del problema primale. Si ricordi che le soluzioni

$$\bar{x}_B = A_B^{-1} b,\ \bar{x}_N = \mathbf{0}, \qquad \bar{u} = A_B^{-\top} c_B$$

sono ottime per i rispettivi problemi.

Analizzeremo due situazioni: nella prima perturberemo una componente del termine noto (del primale), nella seconda perturberemo un coefficiente della funzione obiettivo.[2] In entrambi i casi ci chiederemo per quali perturbazioni la base B resta ottima.

1.3.1 Perturbazione del termine noto

Supponiamo che la i-esima componente del termine noto venga perturbata di una quantità ε e definiamo il termine noto perturbato $b(\varepsilon) = b + \varepsilon e_i$, dove e_i è l'i-esimo vettore della base canonica. Ci proponiamo di scoprire per quali valori $\varepsilon \in \mathbb{R}$ la base B è ancora una base ottima.[3] Come osservato nella Sezione 1.2.2, questo permette anche di stabilire per quali perturbazioni di b_i la variabile duale $\bar{u}_i$ può essere interpretata come prezzo di equilibrio.

[2] Perturbazioni delle componenti della matrice A sono più rare, in quanto la matrice dei vincoli di solito dipende da proprietà strutturali del problema, difficilmente soggette a variazioni.

[3] Si noti che anche se la *base* ottima è la stessa, questo non significa che la *soluzione* ottima resti la stessa, in quanto la formula $x_B = A_B^{-1} b$ dipende, oltre che dalla base B, anche dal termine noto b. Tuttavia, sapere che la base ottima è la stessa ci permette di calcolare la nuova soluzione ottima senza risolvere nuovamente il programma lineare, usando appunto tale formula.

Sia $\bar{x}(\varepsilon)$ la soluzione del problema perturbato associata alla base B: $\bar{x}_B(\varepsilon) = A_B^{-1}b(\varepsilon)$, $\bar{x}_N(\varepsilon) = \mathbf{0}$. Sia inoltre $\bar{u}(\varepsilon)$ la corrispondente soluzione duale: $\bar{u}(\varepsilon) = A_B^{-\top}c_B$. Per il Corollario 1.4, le soluzioni $\bar{x}(\varepsilon), \bar{u}(\varepsilon)$ sono ottime per i rispettivi problemi se e solo se esse sono entrambe ammissibili e $c^\top\bar{x}(\varepsilon) = b(\varepsilon)^\top\bar{u}(\varepsilon)$.

L'ammissibilità duale di $\bar{u}(\varepsilon)$ è ovvia, dato che $\bar{u}(\varepsilon) = \bar{u}$ e i vincoli del duale non sono cambiati. Anche la condizione $c^\top\bar{x}(\varepsilon) = b(\varepsilon)^\top\bar{u}(\varepsilon)$ è certamente verificata, poiché

$$c^\top\bar{x}(\varepsilon) = c^\top A_B^{-1}b(\varepsilon) = \bar{u}(\varepsilon)^\top b(\varepsilon) = b(\varepsilon)^\top\bar{u}(\varepsilon).$$

Resta da imporre l'ammissibilità primale di $\bar{x}(\varepsilon)$. La soluzione $\bar{x}(\varepsilon)$ è ammissibile se e solo se

$$\bar{x}_B(\varepsilon) \geq \mathbf{0} \iff A_B^{-1}b(\varepsilon) \geq \mathbf{0} \iff A_B^{-1}(b + \varepsilon e_i) \geq \mathbf{0} \iff \varepsilon A_B^{-1}e_i \geq -A_B^{-1}b,$$

condizione che può anche essere scritta come $\varepsilon A_B^{-1}e_i \geq -\bar{x}_B$. Poiché $A_B^{-1}e_i$ e $\bar{x}_B$ sono vettori colonna con m componenti, abbiamo ottenuto un sistema di m disequazioni lineari nella sola incognita ε. L'insieme delle soluzioni di questo sistema è necessariamente un intervallo (eventualmente illimitato superiormente e/o inferiormente). Inoltre, tale intervallo deve contenere lo 0, dato che B è per ipotesi una base ottima del problema non perturbato.

Possiamo dunque concludere che la base B resta ottima fintantoché il valore di ε cade dentro un certo intervallo contenente lo 0, dove tale intervallo può essere ricavato risolvendo (rispetto all'incognita ε) il sistema $\varepsilon A_B^{-1}e_i \geq -\bar{x}_B$.

1.3.2 Perturbazione della funzione obiettivo

Supponiamo ora che la j-esima componente di c venga perturbata. Assumiamo $j \in B$ (il caso $j \in N$ può essere trattato in modo simile). Dunque definiamo il vettore perturbato $c(\varepsilon)$ come segue: $c_B(\varepsilon) = c_B + \varepsilon e_j$, $c_N(\varepsilon) = c_N$. Analogamente al caso precedente, ci chiediamo per quali valori $\varepsilon \in \mathbb{R}$ la base B è ancora ottima.[4]

La soluzione primale del problema perturbato associata alla base B è ora $\bar{x}(\varepsilon) = \bar{x}$, mentre la corrispondente soluzione duale è $\bar{u}(\varepsilon) = A_B^{-\top}c_B(\varepsilon)$. Vogliamo di nuovo applicare il Corollario 1.4.

L'ammissibilità primale di $\bar{x}(\varepsilon)$ è ovvia, dato che $\bar{x}(\varepsilon) = \bar{x}$ e i vincoli del primale non sono cambiati. Anche la condizione $c(\varepsilon)^\top\bar{x}(\varepsilon) = b^\top\bar{u}(\varepsilon)$ è certamente verificata, poiché

$$c(\varepsilon)^\top\bar{x}(\varepsilon) = c_B(\varepsilon)^\top\bar{x}_B(\varepsilon) + c_N^\top\bar{x}_N(\varepsilon) = c_B(\varepsilon)^\top A_B^{-1}b = \bar{u}(\varepsilon)^\top b = b^\top\bar{u}(\varepsilon).$$

Resta da imporre l'ammissibilità duale di $\bar{u}(\varepsilon)$. Si noti che il sistema $A^\top u \geq c(\varepsilon)$, che definisce la regione ammissibile del problema duale perturbato, può essere così decomposto:

$$\begin{cases} A_B^\top u \geq c_B(\varepsilon) \\ A_N^\top u \geq c_N. \end{cases}$$

[4] Si noti che in questo caso se la base B resta ottima allora anche la soluzione ottima primale rimane la stessa, in quanto l'espressione $x_B = A_B^{-1}b$ non dipende da c.

Il primo sottosistema è certamente verificato da $\bar{u}(\varepsilon)$, dato che $A_B^\top \bar{u}(\varepsilon) = A_B^\top A_B^{-\top} c_B(\varepsilon) = c_B(\varepsilon)$. Dunque $\bar{u}(\varepsilon)$ è ammissibile se e solo se

$$A_N^\top \bar{u}(\varepsilon) \geq c_N \iff A_N^\top A_B^{-\top} c_B(\varepsilon) \geq c_N \iff A_N^\top A_B^{-\top}(c_B + \varepsilon e_j) \geq c_N$$
$$\iff \varepsilon A_N^\top A_B^{-\top} e_j \geq c_N - A_N^\top A_B^{-\top} c_B.$$

Abbiamo nuovamente ottenuto un sistema di disequazioni lineari nella sola incognita ε, dunque la soluzione è data da un intervallo contenente lo 0.

1.4 Alcuni modelli di Programmazione Lineare

In questa sezione verranno illustrati alcuni modelli classici di programmazione lineare, ma anche alcune varianti un po' più complesse. L'obiettivo è imparare a passare dalla formulazione verbale del problema ad un adeguato modello matematico. Per i casi più semplici verrà anche data l'interpretazione economica del duale.

1.4.1 Ripartizione di risorse

Cominciamo con uno dei modelli più classici. Abbiamo a disposizione m risorse da utilizzare per la realizzazione di n tipologie di prodotti. Per ogni risorsa, è nota la quantità disponibile; per ogni tipologia di prodotto, è noto il profitto derivante dalla vendita; inoltre, per ogni tipologia di prodotto si sa anche quale quantità di ciascuna risorsa è necessaria per la realizzazione di una unità di prodotto (potremmo dire che è nota la "ricetta" del prodotto). Si chiede di utilizzare le risorse in modo da massimizzare il profitto totale, assumendo che il mercato sia sempre in grado di assorbire completamente la merce prodotta.

Indicizziamo con $i = 1, \ldots, m$ le m risorse e con $j = 1, \ldots, n$ le n tipologie di prodotto. Per ogni i, usiamo b_i per indicare la quantità di risorsa i disponibile; per ogni j, indichiamo con c_j il profitto derivante dalla vendita di una unità di prodotto j; infine indichiamo con a_{ij} la quantità di risorsa i necessaria alla realizzazione di una unità di prodotto j.

Passiamo ora alla scelta delle variabili. Appare naturale definire, per ogni j, una variabile x_j che indichi la quantità di prodotto j che si intende realizzare. Il modello sarà allora il seguente:

$$\begin{aligned} \max \quad & c_1 x_1 + \cdots + c_n x_n \\ \text{s.a} \quad & a_{i1} x_1 + \cdots + a_{in} x_n \leq b_i, \quad i = 1, \ldots, m \\ & x_1, \quad \ldots, \quad x_n \geq 0. \end{aligned}$$

La funzione obiettivo esprime il profitto totale. I vincoli assicurano che di ogni risorsa i venga usata al massimo una quantità b_i per realizzare i vari prodotti. Si noti che è necessario includere nel modello anche le condizioni che impongono la non-negatività delle variabili, in quanto una produzione negativa non ha senso.

Interpretazione economica del duale

Il duale del programma lineare dato sopra è

$$\min \quad b_1u_1 + \cdots + \ b_mu_m \tag{1.15}$$

$$\text{s.a } a_{1j}u_1 + \cdots + a_{mj}u_m \geq c_j, \quad j = 1, \ldots, n \tag{1.16}$$

$$u_1, \ \ \ldots, \ \ u_m \geq 0. \tag{1.17}$$

Mostriamo ora come tale programma lineare rappresenti una situazione di mercato ombra, come accennato nella Sezione 1.2.1.

Immaginiamo che qualcuno si offra di acquistare le nostre risorse: il problema consiste dunque nello stabilire il prezzo di ciascuna risorsa. Indichiamo con u_i il prezzo che l'acquirente è disposto a pagare per ogni unità di risorsa i. Il costo totale per l'acquisto delle risorse ammonta allora a $b_1u_1 + \cdots + b_mu_m$, che è esattamente l'espressione che il duale tenta di minimizzare. Naturalmente, affinché la proposta possa suscitare il nostro interesse, l'offerta dell'acquirente dovrà essere in qualche modo concorrenziale. In altre parole, accetteremo di vendere le nostre risorse solo se questa operazione ci frutterà un guadagno maggiore di quello che otterremmo usando le risorse per produrre e vendere i prodotti. Dunque riterremo l'offerta interessante se, per ogni tipologia di prodotto j, ricaveremmo di più (o almeno tanto quanto) dalla vendita delle risorse necessarie a produrre una unità di prodotto j piuttosto che dall'utilizzo delle risorse stesse per la produzione e vendita di una unità del prodotto in questione: questa è esattamente la condizione imposta dal vincolo (1.16). Dunque il duale è il problema che si pone l'acquirente: decidere prezzi di acquisto per le risorse in modo che l'offerta sia conveniente per noi (altrimenti noi rifiuteremmo certamente) e il costo totale per l'acquisto sia il più piccolo possibile.

Si noti che il Teorema di Dualità Debole in questo caso afferma un concetto molto naturale: una qualunque offerta concorrenziale propostaci dall'acquirente ci farebbe guadagnare di più che vendere i prodotti realizzati secondo un qualunque piano produttivo (non solo quello ottimale). L'ottimalità si ha quando la scelta tra le due opzioni è indifferente (Teorema di Dualità Forte).

Infine, come visto nella Sezione 1.2.2, il valore di ciascuna variabile u_i all'ottimo indica di quanto varierebbe il valore ottimo del primale (cioè il nostro profitto) se disponessimo di una unità in più di risorsa i (a patto che la base ottima resti la stessa). In altre parole, se qualcuno ci proponesse di venderci un'ulteriore unità di risorsa i, dato che il nostro profitto aumenterebbe di u_i, noi saremmo disposti a pagare al massimo u_i per avere questa unità in più.

1.4.2 Problema della dieta

Un altro problema classico è il cosiddetto *problema della dieta*. Supponiamo di possedere un allevamento, i cui animali devono ricevere ogni giorno vari nutrienti, che indicizziamo con $i = 1, \ldots, m$ (ad esempio proteine, carboidrati, ecc.). Per ogni nutriente i, il fabbisogno giornaliero totale degli animali dell'allevamento è b_i. I negozi specializzati offrono n prodotti già pronti, contenenti ciascuno una certa quantità di tutti i nutrienti (diversa da prodotto a prodotto),

ma nessuno di questi prodotti offre il giusto mix di nutrienti di cui il nostro allevamento ha bisogno. Supponiamo che ciascuna unità di prodotto $j = 1, \ldots, n$ contenga una quantità a_{ij} di nutriente i. Sia inoltre c_j il costo di una unità di prodotto j. L'obiettivo è decidere in che quantità acquistare i vari prodotti, in modo da minimizzare la spesa ed essere in grado di fornire agli animali tutti i nutrienti di cui hanno bisogno (miscelando opportunamente i prodotti acquistati).

Anche in questo problema la scelta delle variabili è naturale: per ogni j, x_j indicherà la quantità di prodotto j da acquistare. Il modello è il seguente:

$$\begin{aligned} \min \quad & c_1 x_1 + \cdots + c_n x_n \\ \text{s.a} \quad & a_{i1} x_1 + \cdots + a_{in} x_n \geq b_i, \quad j = 1, \ldots, n \\ & x_1, \quad \ldots, \quad x_n \geq 0. \end{aligned}$$

I vincoli assicurano che la quantità di nutriente i che si riesce a ricavare dalla totalità dei prodotti acquistati sia almeno b_i.

Interpretazione economica del duale

Il duale del programma lineare dato sopra è

$$\begin{aligned} \max \quad & b_1 u_1 + \cdots + b_m u_m \\ \text{s.a} \quad & a_{1j} u_1 + \cdots + a_{mj} u_m \leq c_j, \quad j = 1, \ldots, n \\ & u_1, \quad \ldots, \quad u_m \geq 0. \end{aligned}$$

In questo caso il mercato ombra è rappresentato da un offerente che ci propone di venderci direttamente i vari nutrienti non miscelati. Il prezzo proposto per unità di nutriente i è rappresentato dalla variabile u_i. I vincoli assicurano che l'offerta sia interessante per noi: per ogni j (cioè per ogni prodotto che potremmo acquistare nei negozi specializzati), la cifra che pagheremmo all'offerente per avere lo stesso mix di nutrienti che potremmo trovare in una unità di prodotto j non può superare il costo di una unità di tale prodotto (altrimenti ci converrebbe comprare il prodotto j). La funzione obiettivo ovviamente rispecchia il fatto che l'offerente cerca di massimizzare il suo profitto.

Per quanto riguarda i vari teoremi relativi alla dualità, valgono considerazioni analoghe a quelle dell'esempio precedente.

1.4.3 Problema dei traspoti

Dati m depositi (che indicizziamo con $i = 1, \ldots, m$) ed n magazzini di destinazione ($j = 1, \ldots, n$), dobbiamo pianificare il trasporto di una certa merce dai depositi ai magazzini. In ogni deposito i la quantità di merce disponibile è un valore noto a_i; ogni magazzino j ha una domanda pari a b_j. Il costo per trasportare una unità di merce dal deposito i al magazzino j è c_{ij}. Si deve decidere come soddisfare la domanda di tutti i magazzini al costo minimo.

Per ogni coppia (i, j), definiamo una variabile x_{ij} che indicherà la quantità di merce che spediamo dal deposito i al deposito j. Dovremo scrivere due famiglie di vincoli: il primo gruppo di vincoli imporrà la condizione che da ciascun

deposito preleviamo una quantità di merce che non ecceda quella disponibile; la seconda classe di vincoli servirà a garantire che ogni magazzino veda la propria domanda soddisfatta. Il modello è allora il seguente:

$$\begin{aligned} \min \ & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.a} \ & \sum_{j=1}^{n} x_{ij} \leq a_i, \quad i = 1,\dots,m \\ & \sum_{i=1}^{m} x_{ij} \geq b_j, \quad j = 1,\dots,n \\ & x_{ij} \geq 0, \quad i = 1,\dots,m,\ j = 1,\dots,n. \end{aligned}$$

Interpretazione economica del duale

Per comprendere meglio l'interpretazione del programma lineare duale, conviene riscrivere il primale come segue:

$$\min \ \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1.18}$$

$$\text{s.a} \ -\sum_{j=1}^{n} x_{ij} \geq -a_i, \quad i = 1,\dots,m \tag{1.19}$$

$$\sum_{i=1}^{m} x_{ij} \geq b_j, \quad j = 1,\dots,n \tag{1.20}$$

$$x_{ij} \geq 0, \quad i = 1,\dots,m,\ j = 1,\dots,n. \tag{1.21}$$

Associando variabili duali u_i $(i = 1,\dots,m)$ ai vincoli (1.19) e v_j $(j = 1,\dots,n)$ ai vincoli (1.20), il duale si scrive così:

$$\begin{aligned} \max \ & -\sum_{i=1}^{m} a_i u_i + \sum_{j=1}^{n} b_j v_j \\ \text{s.a} \ & -u_i + v_j \leq c_{ij}, \quad i = 1,\dots,m,\ j = 1,\dots,n \\ & u_i \geq 0, \quad i = 1,\dots,m \\ & v_j \geq 0, \quad j = 1,\dots,n. \end{aligned}$$

Il mercato ombra è qui rappresentato da un soggetto che si offre di acquistare tutta la merce in giacenza nei nostri depositi e di rivendercela nei magazzini di destinazione, risparmiandoci così il compito di effettuare il trasporto. Il prezzo unitario a cui l'offerente acquista la merce nel deposito i è u_i, il prezzo unitario a cui ce la rivende nel magazzino j è v_j. Come sempre, i vincoli duali assicurano che l'offerta sia per noi interessante: per ogni deposito i e magazzino j, il bilancio risultante dalla vendita di una unità di merce nel deposito i e dal suo riacquisto nel magazzino j non deve eccedere quello che spenderemmo per effettuare noi stessi il trasporto di tale unità di merce. La funzione obiettivo

rispecchia il fatto che l'offerente cerca di massimizzare il suo profitto, che è dato dalla spesa totale per acquistare la merce nei depositi meno il ricavo ottenuto vendendola nei magazzini di destinazione.

1.4.4 Problemi di produzione con gestione delle scorte

Dobbiamo pianificare la produzione di una certa merce per un orizzonte temporale di n periodi (per esempio settimane o mesi), che indicizziamo con $t = 1, \ldots, n$. Assumiamo per ora che la domanda che dobbiamo soddisfare in ogni periodo t sia nota a priori: chiamiamola d_t. Produrre la merce in un periodo piuttosto che in un altro non ha necessariamente lo stesso costo: indichiamo con p_t il costo di produzione di una unità di merce nel periodo t. Disponiamo inoltre di un magazzino: il costo per tenere in magazzino una unità di merce dal periodo t al periodo $t+1$ è indicato con h_t. Si chiede di stabilire quando e quanto produrre (e di conseguenza come utilizzare il magazzino) in modo da soddisfare tutte le domande e minimizzare il costo totale. (Non includiamo in questo modello il ricavo derivante dalla vendita della merce, in quanto stiamo assumendo che la domanda sia nota a priori, quindi il ricavo è anch'esso noto a priori: dunque massimizzare il guadagno, definito come ricavo meno costi, equivale a minimizzare i costi.)

Per ogni periodo t, conviene indicare con x_t la quantità prodotta nel periodo t e con s_t la quantità di merce in eccedenza alla fine del periodo t (chiamata *stock*), che dovrà essere immagazzinata. In ciasun periodo deve valere una sorta di "principio di conservazione": la quantità di merce "entrante" nel periodo (data dalla quantità prodotta più la quantità giacente in magazzino "proveniente" dal periodo precedente) deve essere uguale alla quantità "uscente" (data dalla domanda che dobbiamo soddisfare più lo stock che verrà "mandato" al periodo successivo). La Figura 1.1 può aiutare a visualizzare il concetto. Dovremo allora scrivere il seguente modello:

$$\min \quad \sum_{t=1}^{n} (p_t x_t + h_t s_t) \tag{1.22}$$

$$\text{s.a} \quad s_{t-1} + x_t = d_t + s_t, \quad t = 1, \ldots, n \tag{1.23}$$

$$x_t \geq 0,\ s_t \geq 0, \qquad t = 1, \ldots, n, \tag{1.24}$$

dove si intende che $s_0 = 0$. Si noti che la presenza di equazioni nel modello permetterebbe di eliminare alcune variabili (ad esempio tutte le variabili s_t oppure tutte le variabili x_t). Ciononostante, mantenere tutte le variabili rende il modello più intuitivo e permette di interpretare con immediatezza il significato della soluzione ottima.

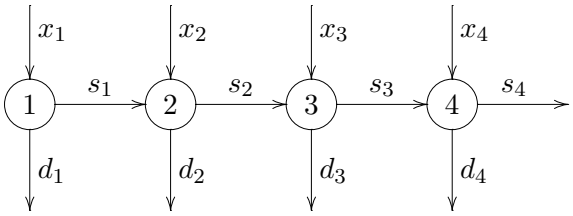

Figura 1.1: Illustrazione del "principio di conservazione della merce" e relativi vincoli (1.23). I numeri dentro i cerchi indicano i periodi.

Interpretazione economica del duale

Il duale del programma lineare dato sopra è

$$\max \quad \sum_{t=1}^{n} d_t u_t \tag{1.25}$$

$$\text{s.a} \qquad u_t \le p_t, \quad t = 1, \dots, n \tag{1.26}$$

$$u_{t+1} - u_t \le h_t, \quad t = 1, \dots, n, \tag{1.27}$$

dove si intende che $u_{n+1} = 0$.

Per interpretare il duale, possiamo immaginare che qualcuno si offra di venderci in ciascun periodo la merce che vorremmo produrre, evitandoci sia l'onere di produrla noi stessi sia il costo di gestione del magazzino, in quanto acquisteremmo la merce necessaria ogni periodo. Il prezzo proposto per ogni unità di merce nel periodo t è u_t. Come di consueto, ci aspettiamo che i vincoli garantiscano che la proposta sia interessante, cioè che in nessun caso ci convenga produrre da noi o utilizzare il magazzino: in effetti, i vincoli (1.26) impongono la condizione che acquistare la merce dall'offerente non sia più costoso che produrla; e i vincoli (1.27), che conviene riscrivere come $u_t + h_t \ge u_{t+1}$, escludono che sia più conveniente acquistare della merce nel periodo t per poi tenerla in magazzino fino al periodo $t+1$ piuttosto che acquistarla direttamente nel periodo $t+1$. La funzione obiettivo è la somma di denaro necessaria all'acquisto.

Tuttavia, il programma lineare (1.25)–(1.27) presenta un'anomalia che sembrerebbe contrastare con questa interpretazione economica: mancano i vincoli $u_t \ge 0$ per $t = 1, \dots, n$, che invece ci aspetteremmo, visto che u_t è un prezzo. Mostriamo ora che in realtà queste condizioni sono sempre soddisfatte dalle soluzioni ottime e dunque la loro assenza non costituisce un problema.

Osservazione 1.8 *Assumendo che tutti i costi p_t, h_t e le domande d_t siano positivi, se u è una soluzione ottima di* (1.25)–(1.27) *allora $u \ge \mathbf{0}$.*

Dimostrazione. Sia u una soluzione ottima ed assumiamo per assurdo che $u_\tau < 0$ per qualche τ. Scegliamo il minimo indice τ con questa proprietà (cioè $u_t \ge 0$ per ogni $t = 1, \dots, \tau - 1$). Definiamo un vettore u' ponendo

$$u'_t = \begin{cases} u_t & \text{se } t \ne \tau \\ 0 & \text{se } t = \tau. \end{cases}$$

Verifichiamo che u' è una soluzione ammissibile. I vincoli $u'_t \leq p_t$ sono ovviamente soddisfatti per ogni $t \neq \tau$ (dato che $u'_t = u_t \leq p_t$) e anche per $t = \tau$ (in quanto $u'_\tau = 0 \leq p_t$). I vincoli $u'_{t+1} - u'_t \leq h_t$ sono certamente soddisfatti per $t \notin \{\tau - 1, \tau\}$, perché in tal caso $u'_{t+1} - u'_t = u_{t+1} - u_t \leq h_t$. Per quanto riguarda il caso $t = \tau - 1$, abbiamo $u'_\tau - u'_{\tau-1} = 0 - u_{\tau-1} \leq 0 \leq h_{\tau-1}$. E per il caso $t = \tau$, abbiamo $u'_{\tau+1} - u'_\tau = u_{\tau+1} - 0 < u_{\tau+1} - u_\tau \leq h_t$.

Dunque u' è una soluzione ammissibile. Poiché il costo di u' è $\sum_{t=1}^n d_t u'_t < \sum_{t=1}^n d_t u_t$, abbiamo una contraddizione all'ottimalità di u. □

Variante: domanda elastica

Consideriamo ora la variante del problema in cui la domanda da soddisfare in ciascun periodo t non ha un valore fisso, ma è compresa tra un valore minimo d_t ed un valore massimo D_t. Più precisamente, assumiamo di essere tenuti a soddisfare una domanda di almeno d_t e che il mercato sia in grado di assorbire una domanda massima D_t. In questa situazione, piuttosto che minimizzare i costi vorremo massimizzare il profitto, definito come differenza tra ricavo e costi. Indichiamo con r_t il ricavo ottenuto dalla vendita di una unità di merce nel periodo t.

Le variabili sono le stesse di prima, ma ora usiamo anche variabili v_t per indicare la quantità venduta nel periodo t. Il modello è sostanzialmente invariato, ma la domanda, che prima era nota, ora va sostituita con la variabile che indica la quantità venduta (alla quale vanno imposti limite inferiore e limite superiore):

$$\begin{aligned} \max \quad & \sum_{t=1}^{n} (r_t v_t - p_t x_t - h_t s_t) \\ \text{s.a} \quad & s_{t-1} + x_t = v_t + s_t, \qquad t = 1, \ldots, n \\ & d_t \leq v_t \leq D_t, \qquad t = 1, \ldots, n \\ & x_t \geq 0,\ s_t \geq 0, \qquad t = 1, \ldots, n. \end{aligned}$$

Variante: modello con backlogging

Esistono numerosissime altre varianti del modello presentato sopra, alcune delle quali verranno illustrate più avanti in questo e nel prossimo capitolo. A titolo di esempio, ne vediamo ora una particolarmente importante: ammettiamo la possibilità di effettuare *backlogging*, cioè di consegnare (vendere) la merce anche in un periodo successivo rispetto a quello in cui ci era stata richiesta. Ovviamente questo comporterà una penalità (ad esempio saremo costretti a fare uno sconto): indichiamo con q_t la penalità per una unità di merce prodotta nel periodo $t + 1$ ma usata per soddisfare la domanda (o parte di essa) del periodo precedente.

Usiamo le stesse variabili di prima, più le variabili b_t per indicare la quantità di merce prodotta nel periodo $t + 1$ ed usata per soddisfare la domanda del periodo precedente. Dovremo solo cambiare i vincoli di conservazione della

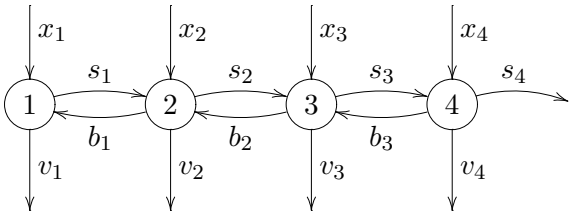

Figura 1.2: Illustrazione del "principio di conservazione della merce" e relativi vincoli (1.29) per un problema di produzione con backlogging. I numeri dentro i cerchi indicano i periodi.

merce, facendoci aiutare dalla Figura 1.2:

$$\max \quad \sum_{t=1}^{n}(r_t v_t - p_t x_t - h_t s_t - q_t b_t) \tag{1.28}$$

$$\text{s.a} \quad s_{t-1} + x_t + b_t = v_t + s_t + b_{t-1}, \quad t = 1, \ldots, n \tag{1.29}$$

$$d_t \leq v_t \leq D_t, \qquad t = 1, \ldots, n \tag{1.30}$$

$$x_t \geq 0,\ s_t \geq 0,\ b_t \geq 0, \qquad t = 1, \ldots, n, \tag{1.31}$$

dove si intende $s_0 = b_0 = b_n = 0$.

1.4.5 Produzione di più prodotti su una macchina

Supponiamo di disporre di una macchina per la produzione di m articoli (indicizzati con $i = 1, \ldots, m$). Se la macchina viene utilizzata per produrre l'articolo i, il tasso produttivo è di ρ_i unità al giorno, mentre il costo unitario di produzione è p_i. Per ogni articolo c'è una domanda massima d_i (eventuali pezzi in eccesso non possono essere venduti) e il prezzo di vendita è r_i. Si vuole pianificare l'utilizzo della macchina per i prossimi N giorni lavorativi, in modo da massimizzare il profitto.

In questo problema ci sono almeno due scelte ugualmente buone per le variabili: possiamo definire t_i come il tempo (espresso in giornate lavorative) in cui la macchina verrà destinata alla produzione di articoli di tipo i, oppure x_i come la quantità prodotta. Le due grandezze sono legate tra loro dalla relazione $x_i = t_i \rho_i$. (Si noti che non ha alcuna importanza determinare *quando* la macchina sarà destinata alla produzione di un articolo piuttosto che un altro: in questo problema conta solo il *quanto*.) Scegliendo (senza particolare motivo) di usare le variabili x_i, il modello può essere scritto così:

$$\begin{aligned} \max \ & \sum_{i=1}^{m}(r_i - p_i)x_i \\ \text{s.a} \ & \sum_{i=1}^{m} \frac{x_i}{\rho_i} \leq N \\ & 0 \leq x_i \leq d_i. \end{aligned}$$

Variante: più periodi e magazzino

Consideriamo lo stesso problema, ma su più periodi e con un magazzino da gestire. Ci sono n periodi, ciascuno con N_t giorni lavorativi ($t = 1, \ldots, n$). La domanda massima varia sia a seconda dell'articolo che del periodo, ed è indicata con d_{it}. Stesso discorso per i costi di produzione unitari p_{it}, i prezzi di vendita r_{it} e il costo di magazzino h_{it} (per conservare una unità di articolo i dal periodo t al successivo). Assumiamo invece che il tasso di produzione ρ_i dipenda solo dal tipo di articolo (ma cambierebbe ben poco se dipendesse anche dal periodo). Infine, il magazzino ha una capacità massima di C unità.

Qui useremo variabili x_{it} per la quantità di articoli di tipo i prodotti nel periodo t, s_{it} per la quantità di articoli di tipo i in stock alla fine del periodo t e v_{it} per la quantità di articoli di tipo i venduti nel periodo t. Il modello sarà il seguente:

$$\max \quad \sum_{i=1}^{m}\sum_{t=1}^{n}(r_{it}v_{it} - p_{it}x_{it} - h_{it}s_{it}) \tag{1.32}$$

$$\text{s.a} \qquad \sum_{i=1}^{m}\frac{x_{it}}{\rho_i} \le N_t, \qquad t = 1, \ldots, n \tag{1.33}$$

$$s_{i,t-1} + x_{it} = v_{it} + s_{it}, \qquad i = 1, \ldots, m,\ t = 1, \ldots, n \tag{1.34}$$

$$\sum_{i=1}^{m} s_{it} \le C, \qquad t = 1, \ldots, n \tag{1.35}$$

$$0 \le v_{it} \le d_{it}, \qquad i = 1, \ldots, m,\ t = 1, \ldots, n \tag{1.36}$$

$$x_{it} \ge 0,\ s_{it} \ge 0, \qquad i = 1, \ldots, m,\ t = 1, \ldots, n. \tag{1.37}$$

Il vincolo (1.33) impone che in ogni periodo la macchina venga utilizzata al massimo per il tempo prestabilito di N_t giorni. Le equazioni (1.34) assicurano che la merce di ciascun tipo i "si conservi" nei vari periodi. Le disequazioni (1.35) garantiscono che in ogni periodo venga rispettata la capienza massima del magazzino (dunque è necessario sommare su tutte le tipologie di articolo). I vincoli restanti sono di ovvia interpretazione.

Variante: più macchine

Assumiamo ora che ci siano più macchine, diciamo k. Per ogni macchina $j = 1, \ldots, k$ e per ogni tipo di prodotto $i = 1, \ldots, m$, sia ρ_{ij} il tasso di produzione della macchina j quando viene impiegata per produrre articoli di tipo i. Il resto del problema è invariato.

Le variabili saranno le stesse di prima, tranne quelle che indicano la quantità prodotta: ora useremo x_{ijt} per indicare la quantità di merce di tipo i prodotta

nel periodo t utilizzando la macchina j. Otteniamo allora il seguente modello:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{m}\sum_{t=1}^{n}\Big(r_{it}v_{it} - p_{it}\sum_{j=1}^{k}x_{ijt} - h_{it}s_{it}\Big) \\
\text{s.a} \quad & \sum_{i=1}^{m}\frac{x_{ijt}}{\rho_{ij}} \le N_t, && j=1,\dots,k,\ t=1,\dots,n \\
& s_{i,t-1} + \sum_{j=1}^{k}x_{ijt} = v_{it} + s_{it}, && i=1,\dots,m,\ t=1,\dots,n \\
& \sum_{i=1}^{m}s_{it} \le C, && t=1,\dots,n \\
& 0 \le v_{it} \le d_{it}, && i=1,\dots,m,\ t=1,\dots,n \\
& x_{ijt} \ge 0,\ s_{it} \ge 0, && i=1,\dots,m,\ j=1,\dots,n,\ t=1,\dots,n.
\end{aligned}
$$

Le modifiche rispetto al modello precedente sono solo due: la prima famiglia di vincoli (che limita l'utilizzo delle macchine) va ovviamente scritta per ogni macchina (cioè per ogni j); inoltre, nella funzione obiettivo e nel vincolo sulla capacità del magazzino la merce di tipo i prodotta al tempo t è espressa dalla somma $\sum_{j=1}^{k} x_{ijt}$, in quanto dobbiamo tenere conto dei pezzi prodotti da tutte le macchine.

1.4.6 Pianificazione di turni

Dobbiamo pianificare i turni dei lavoratori di un certo reparto. I turni devono ripetersi identici ogni settimana, quindi sarà sufficiente pianificarli per una singola settimana. Ogni turno consiste di una sequenza di 5 giorni di lavoro consecutivi, seguiti da 2 giorni di riposo (sono dunque possibili 7 tipi di turno diversi). Il numero di lavoratori necessari al corretto funzionamento del reparto varia a seconda del giorno della settimana ed è indicato da un vettore b con 7 componenti. Si chiede di pianificare i turni in modo che ogni giorno sia garantito il numero minimo di lavoratori necessari, minimizzando il numero di lavoratori complessivamente in servizio. (Un turno pieno consta di un certo numero di ore di lavoro al giorno, ma ammettiamo anche la presenza di lavoratori part-time.)

Indichiamo con $x_1, x_2, \dots, x_7$ il numero di lavoratori che iniziano il turno rispettivamente di lunedì, martedì, ..., domenica. Un valore non intero delle componenti di x nella soluzione ottima, ad esempio $x_1 = 25.4$, verrà interpretato così: 25 lavoratori assunti full-time più un lavoratore part-time (che ogni giorno lavora per un tempo pari al 40% di un turno pieno) iniziano il turno di lunedì.

Il modello è il seguente:

$$
\begin{array}{rl}
\min & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \\
\text{s.a} & x_1 + x_4 + x_5 + x_6 + x_7 \geq b_1 \\
& x_1 + x_2 + x_5 + x_6 + x_7 \geq b_2 \\
& x_1 + x_2 + x_3 + x_6 + x_7 \geq b_3 \\
& x_1 + x_2 + x_3 + x_4 + x_7 \geq b_4 \\
& x_1 + x_2 + x_3 + x_4 + x_5 \geq b_5 \\
& x_2 + x_3 + x_4 + x_5 + x_6 \geq b_6 \\
& x_3 + x_4 + x_5 + x_6 + x_7 \geq b_7 \\
& x_1, \quad x_2, \quad x_3, \quad x_4, \quad x_5, \quad x_6, \quad x_7 \geq 0.
\end{array}
$$

Capitolo 2

Programmazione Lineare Intera

2.1 Introduzione

Un *programma lineare intero* è un problema di ottimizzazione vincolata in cui la funzione obiettivo è lineare e la regione ammissibile è costituita dall'insieme dei vettori interi[1] che soddisfano un dato sistema di equazioni e/o disequazioni lineari. Se scriviamo un sistema di equazioni e/o disequazioni lineari nella forma compatta $Ax \sim b$, dove $A \in \mathbb{R}^{m \times n}$ e $b \in \mathbb{R}^m$, un programma lineare intero è dunque un problema del tipo

$$\begin{aligned} \max\ & c^\top x \\ \text{s.a}\ & Ax \sim b \\ & x \in \mathbb{Z}^n, \end{aligned}$$

dove $c \in \mathbb{R}^n$.

Un *programma lineare intero misto* è la variante in cui la condizione di interezza è richiesta solo per un certo sottoinsieme delle variabili:

$$\begin{aligned} \max\ & c^\top x \\ \text{s.a}\ & Ax \sim b \\ & x_i \in \mathbb{Z}, \quad i \in I, \end{aligned}$$

per un qualche assegnato sottoinsieme $I \subseteq \{1, \ldots, n\}$. Le variabili x_i con $i \in I$ sono chiamate *variabili intere*, le altre *variabili continue*. A volte, per marcare la distinzione tra i due problemi, un programma lineare intero viene chiamato *programma lineare intero puro*.

Di fondamentale importanza in programmazione lineare intera (mista) sono le *variabili binarie*, cioè quelle che possono assumere solo valore 0 o 1: come vedremo negli esempi, tali variabili vengono spesso usate per descrivere decisioni del tipo sì/no. Si noti che un programma lineare intero (misto) permette sempre di imporre la condizione che una certa variabile x_i sia binaria scrivendo i vincoli

[1] Un vettore è detto *intero* se tutte le sue componenti sono intere.

$0 \leq x_i \leq 1$, $x_i \in \mathbb{Z}$. Per brevità, di norma scriveremo questi vincoli direttamente nella forma $x_i \in \{0, 1\}$.

2.2 Modelli e tecniche di modellizzazione

In questa sezione verranno illustrati alcuni modelli di programmazione lineare intera (mista) ed alcune tecniche di modellizzazione di particolare interesse.

Un primo ovvio utilizzo delle variabili intere si può avere ad esempio in problemi di produzione in cui la merce o le risorse sono indivisibili: in questo caso basterà aggiungere ai modelli del capitolo precedente i vincoli che impongono l'interezza delle variabili. Allo stesso modo, nel problema di pianificazione dei turni possiamo richiedere l'interezza delle variabili se non sono ammesse assunzioni part-time. Tuttavia, vedremo che la potenza delle variabili intere (in particolare delle variabili binarie) va ben oltre questa semplice estensione.

Si tenga presente che i modelli presentati, anche se in alcuni casi vengono introdotti per problemi apparentemente molto specifici, illustrano delle tecniche di modellizzazione che sono di amplissimo utilizzo in programmazione lineare intera (mista)

2.2.1 Problema dello zaino

Sono dati n oggetti (indicizzati con $i = 1, \ldots, n$), ciascuno con un peso p_i ed un'utilità u_i. Si chiede di selezionare un sottoinsieme di oggetti in modo tale che il loro peso totale non ecceda un dato valore P (la portata dello zaino) e l'utilità totale sia massimizzata.

Per ogni $i = 1, \ldots, n$, definiamo una variabile binaria x_i il cui valore è 1 se e solo se l'oggetto i viene selezionato. In questo modo l'espressione $p_i x_i$ dà il peso dell'oggetto i se l'oggetto viene selezionato e 0 altrimenti (analogamente per l'utilità). Possiamo allora formulare il problema nel modo seguente:

$$\begin{aligned} \max \ & \sum_{i=1}^{n} u_i x_i \\ \text{s.a} \ & \sum_{i=1}^{n} p_i x_i \leq P \\ & x_i \in \{0, 1\}, \quad i = 1, \ldots, n. \end{aligned}$$

Il problema dello zaino permette di modellizzare anche problemi che a prima vista potrebbero sembrare di natura totalmente diversa. Si consideri ad esempio il caso di un investitore, intenzionato ad investire un capitale massimo P, al quale vengono prospettati n possibili "pacchetti", ciascuno dei quali richiede un investimento p_i e dà un guadagno atteso u_i. Il problema di decidere quali pacchetti acquistare senza eccedere la somma prefissata P e massimizzando il guadagno atteso è esattamente il problema dello zaino.

2.2.2 Problema di assegnamento

Dati n lavoratori ($i = 1, \ldots, n$) ed n mansioni da eseguire ($j = 1, \ldots, n$), si deve assegnare ogni mansione ad un lavoratore diverso in modo da minimizzare il costo totale dei compensi da pagare ai lavoratori. Si sa che assegnare la mansione j al lavoratore i comporta il pagamento di un compenso pari a c_{ij}.

Per ogni coppia (i, j) definiamo una variabile binaria x_{ij} che vale 1 se e solo se al lavoratore i viene assegnata la mansione j. Dobbiamo far sì che ad ogni lavoratore sia assegnata un'unica mansione e che ogni mansione sia svolta da un unico lavoratore (nessuno dei due gruppi di vincoli da solo sarebbe sufficiente):

$$\begin{aligned} \min \ & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.a} \ & \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \\ & \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n \\ & x_{ij} \in \{0, 1\}, \quad i, j = 1, \ldots, n. \end{aligned}$$

2.2.3 Scheduling

I problemi di *scheduling* costituiscono una vastissima famiglia di problemi in cui, dati dei lavori da eseguire e una o più macchine su cui è possibile svolgerli, si chiede di stabilire a quale macchina assegnare ciascun lavoro, in modo da minimizzare una qualche funzione obiettivo (che varia da problema a problema). A titolo di esempio, consideriamo qui una delle tante varianti del problema.

Sono dati n lavori ($j = 1, \ldots, n$) ed m macchine ($i = 1, \ldots, m$). Ogni lavoro può essere svolto da una qualunque delle macchine, ma il tempo di esecuzione non è necessariamente lo stesso: indichiamo con t_{ij} il tempo che la macchina i impiega a portare a compimento il lavoro j. Si devono assegnare i lavori alle macchine in modo da minimizzare il *makespan*, ovvero il tempo totale compreso tra l'istante in cui le macchine vengono messe in azione (che sarà lo stesso per tutte le macchine) e l'istante in cui tutti i lavori sono stati portati a compimento.

Questo problema può essere modellizzato tramite il seguente programma lineare intero misto, che discutiamo più sotto:

$$\min \ T \tag{2.1}$$

$$\text{s.a} \ \sum_{i=1}^{m} x_{ij} = 1, \qquad j = 1, \ldots, n \tag{2.2}$$

$$T \geq \sum_{j=1}^{n} t_{ij}x_{ij}, \quad i = 1, \ldots, m \tag{2.3}$$

$$x_{ij} \in \{0, 1\}, \qquad i = 1, \ldots, m, \ j = 1, \ldots, n. \tag{2.4}$$

Per ogni macchina i e lavoro j, c'è una variabile binaria x_{ij} che vale 1 se e solo se il lavoro j viene assegnato alla macchina i. I vincoli (2.2) impongono

che ogni lavoro sia assegnato ad una e una sola macchina. Indichiamo inoltre con una variabile T il makespan, che è la quantità che vogliamo minimizzare (funzione obiettivo (2.1)). Per definizione di makespan, vogliamo che T valga esattamente il massimo dei tempi di attività delle varie macchine. Considerato che il tempo di attività della macchina i è dato dall'espressione $\sum_{j=1}^{n} t_{ij}x_{ij}$, vorremmo imporre la condizione

$$T = \max_{i=1,\dots,m} \sum_{j=1}^{n} t_{ij}x_{ij}. \tag{2.5}$$

Tuttavia questo vincolo non è lineare. Quello che possiamo fare è scrivere i vincoli (2.3), che garantiscono che T valga *almeno* il massimo in (2.5). Ora, considerato che la funzione obiettivo forzerà T a prendere il valore più piccolo possibile, possiamo essere sicuri che la soluzione ottima soddisferà almeno uno dei vincoli (2.3) ad uguaglianza (altrimenti potremmo ottenere una soluzione ammissibile migliore diminuendo leggermente il valore di T, contraddicendo l'ottimalità della soluzione). In altre parole, siamo sicuri che all'ottimo il valore di T sarà esattamente il massimo in (2.5).

È importante mettere in risalto l'idea usata: invece di imporre che T sia esattamente il makespan per tutte le soluzioni *ammissibili* (cosa che appare ardua da scrivere tramite vincoli lineari), ci accontentiamo di far sì che T sia esattamente il makespan per tutte le soluzioni *ottime*. Vedremo che quest'idea viene utilizzata di frequente per ovviare alla non-linearità di certi vincoli.

2.2.4 Lot-sizing

Con *lot-sizing* si indica una famiglia di problemi di pianificazione della produzione caratterizzati dalla presenza di costi fissi (cioè non proporzionali alla quantità di merce). Si tratta sostanzialmente di varianti dei problemi illustrati nella Sezione 1.4.4.

Nel caso più semplice, si deve pianificare la produzione di una certa merce per un orizzonte temporale di n periodi, che indicizziamo con $t = 1, \dots, n$. Per ogni periodo t, la domanda d_t è nota a priori e deve essere soddisfatta completamente (mentre eventuale merce in eccesso non può essere venduta). Indichiamo con p_t il costo di produzione di un'unità di merce nel periodo t e con h_t il costo per tenere in magazzino un'unità di merce dal periodo t al periodo $t + 1$. Inoltre, in ogni periodo t in cui si decida di produrre qualcosa si deve pagare un costo fisso f_t che non dipende dalla quantità prodotta. Assumiamo infine che in ogni periodo t vi sia un limite massimo alla quantità che può essere prodotta (capacità), diciamo C_t. Si chiede di stabilire quando e quanto produrre in modo da soddisfare tutte le domande e minimizzare il costo totale.

Come negli esempi della Sezione 1.4.4, definiamo variabili x_t per la quantità prodotta nel periodo t e s_t per la quantità immagazzinata alla fine del periodo t. Introduciamo inoltre variabili binarie y_t che valgono 1 se e solo se nel periodo t produciamo qualcosa (e dunque siamo tenuti a pagare il costo fisso). In altre parole, vorremmo avere $y_t = 1$ se e solo se $x_t > 0$. Scriviamo il seguente

programma lineare intero misto:

$$\min \quad \sum_{t=1}^{n}(p_t x_t + h_t s_t + f_t y_t) \tag{2.6}$$

$$\text{s.a} \quad s_{t-1} + x_t = d_t + s_t, \qquad t = 1, \ldots, n \tag{2.7}$$

$$x_t \leq C_t y_t, \qquad t = 1, \ldots, n \tag{2.8}$$

$$x_t \geq 0,\ s_t \geq 0,\ y_t \in \{0, 1\}, \quad t = 1, \ldots, n, \tag{2.9}$$

dove $s_0 = 0$.

Il significato della funzione obiettivo è chiaro: oltre ai costi usuali, abbiamo aggiunto i costi fissi f_t, che vengono conteggiati se e solo se la corrispondente variabile y_t vale 1. Il primo gruppo di vincoli impone, come di consueto, la "conservazione della merce". I vincoli (2.8) servono ad assicurare che la variabile y_t valga 1 non appena la quantità prodotta nel periodo t è positiva (cioè $x_t > 0$). In effetti, se $x_t > 0$ allora (2.8) costringe y_t a prendere un valore positivo; e dato che y_t è una variabile binaria, questo valore non può che essere 1. Contemporaneamente, quando $y_t = 1$ lo stesso vincolo impone anche un limite superiore C_t al valore di x_t, come richiesto dal problema. Resta solo un particolare da considerare: i vincoli (2.8) garantiscono che se $x_t > 0$ allora $y_t = 1$ (e $x_t \leq C_t$), ma non assicurano che se $x_t = 0$ allora $y_t = 0$. Tuttavia questo non costituisce un problema, dato che in una soluzione ottima non avremo mai $x_t = 0$ e $y_t = 1$: se così fosse, otterremmo una soluzione più economica diminuendo y_t a zero (assumendo, come naturale, che i costi fissi f_t siano positivi).

Variante: nessuna capacità

Consideriamo la variante del problema precedente in cui non vi è alcun limite massimo alla capacità produttiva. Come scrivere i vincoli (2.8) in questo caso?

L'idea è quella di scegliere i valori C_t in modo che i vincoli (2.8) garantiscano ancora che se $x_t = 1$ allora $y_t = 1$, ma non impongano più un limite massimo a x_t. Si noti che in una soluzione ottima non produrremo mai più della domanda totale, che ammonta a $M = \sum_{t=1}^{n} d_t$. Dunque, se immaginiamo che esista una capacità massima di M in ciascun periodo, il problema resta invariato. Sostituiremo allora i vincoli (2.8) con

$$x_t \leq M y_t, \quad t = 1, \ldots, n.$$

In questo modo, quando $y_t = 1$ la limitazione $x_t \leq M$ diventa del tutto ininfluente.

Variante: produzione in lotti

Altra variante: in ciascun periodo la produzione avviene in lotti di una certa grandezza C; ogni volta che produciamo un nuovo lotto dobbiamo pagare un costo fisso f_t. Ad esempio, se $C = 10$ e produciamo 23 unità, dobbiamo pagare il costo fisso tre volte, anche se il terzo lotto non è di grandezza massima.

In questo caso il modello presenterà un'unica variazione: le variabili y_t, invece che binarie, saranno numeri interi:

$$\min \quad \sum_{t=1}^{n}(p_t x_t + h_t s_t + f_t y_t) \tag{2.10}$$

$$\text{s.a} \quad s_{t-1} + x_t = d_t + s_t, \qquad t = 1, \ldots, n \tag{2.11}$$

$$x_t \leq C y_t, \qquad t = 1, \ldots, n \tag{2.12}$$

$$x_t \geq 0,\ s_t \geq 0,\ y_t \in \mathbb{Z}, \quad t = 1, \ldots, n \tag{2.13}$$

(non è necessario scrivere $y_t \geq 0$ per $t = 1, \ldots, n$, in quanto tali vincoli seguono da (2.12) e dalla non-negatività delle variabili x_t).

Un modo per convincersi della correttezza del modello è notare che il legame che vorremmo imporre tra x_t e y_t è $y_t = \lceil x_t/C \rceil$, dove la notazione $\lceil \cdot \rceil$ indica l'arrotondamento per eccesso. Scrivendo il vincolo (2.12), imponiamo $y_t \geq x_t/C$; ma poiché y_t è un intero, varrà anche $y_t \geq \lceil x_t/C \rceil$. Resta da osservare che nelle soluzioni ottime si avrà necessariamente l'uguaglianza.

Se poi ci fosse anche un limite massimo alla produzione, diciamo k_t lotti per il periodo t, basterebbe aggiungere le disequazioni $y_t \leq k_t$ per $t = 1, \ldots, n$.

Altre varianti

Per quanto riguarda il caso in cui la domanda è elastica e/o vi è possibilità di effettuare backlogging, è immediato adattare i modelli visti nella Sezione 1.4.4.

2.2.5 Facility location

I problemi di *facility location* possono essere visti come un'estensione del problema dei trasporti al caso in cui siano presenti costi fissi. Sono dati n clienti da soddisfare ($j = 1, \ldots, n$), ciascuno con una domanda d_j, ed m centri di distribuzione ($i = 1, \ldots, m$). Il costo unitatio per servire il cliente j utilizzando il centro i è c_{ij}. Vi è inoltre un costo fisso f_i da pagare se si decide di attivare il centro di distribuzione i. Non è obbligatorio attivare tutti i centri di distribuzione: alcuni di essi possono restare chiusi. Nella prima versione che consideriamo, assumiamo che ogni centro i possa soddisfare una domanda totale massima di C_i unità; assumiamo inoltre che la domanda sia indivisibile, cioè che ogni cliente debba vedere la propria domanda soddisfatta da un unico centro di distribuzione. L'obiettivo è quello di servire tutti i clienti minimizzando i costi.

Per ogni i e per ogni j, definiamo una variabile binaria x_{ij} che vale 1 se e solo se il cliente j viene servito dal centro i (la variabile è binaria a causa dell'assunzione che la domanda è indivisibile). Definiamo inoltre, per ogni i, una variabile binaria y_i che vale 1 se e solo se il centro i viene attivato. Possiamo

allora scrivere il seguente modello:

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}d_j x_{ij} + \sum_{i=1}^{m} f_i y_i \tag{2.14}$$

$$\text{s.a} \quad \sum_{i=1}^{m} x_{ij} = 1, \qquad j = 1,\dots,n \tag{2.15}$$

$$\sum_{j=1}^{n} d_j x_{ij} \le C_i y_i, \qquad i = 1,\dots,m \tag{2.16}$$

$$x_{ij}, y_i \in \{0,1\}, \qquad i = 1,\dots,m,\ j = 1,\dots,n. \tag{2.17}$$

Si noti che la quantità $d_j x_{ij}$, che compare sia nella funzione obiettivo che in (2.16), non è altro che la quantità di merce che il cliente j riceve dal centro i. Le equazioni (2.15) garantiscono che ogni cliente sia servito da esattamente uno dei centri. I vincoli (2.16) impongono contemporaneamente due condizioni: primo, se un centro i serve almeno uno dei clienti allora la corrispondente variabile y_i è costretta a valere 1; secondo, quando $y_i = 1$ (cioè il centro i viene attivato) viene imposto un limite massimo di C_i alla quantità di merce che il centro i distribuisce ai vari clienti. Si noti che stiamo accettando anche soluzioni in cui, per un qualche i fissato, si ha $x_{ij} = 0$ per tutti i j e $y_i = 1$, cosa che vorremmo escludere; ciononostante, analogamente a quanto visto per il caso del lot-sizing, questa situazione non può verificarsi in una soluzione ottima (assumendo costi fissi positivi).

Nel caso in cui la domanda sia divisibile, cioè un cliente può essere servito anche da più centri, sarà sufficiente rimpiazzare i vincoli $x_{ij} \in \{0,1\}$ con $0 \le x_{ij} \le 1$ per ogni i ed ogni j: il significato di x_{ij} sarà la frazione di domanda del cliente j che viene soddisfatta dal centro i.

Variante: nessuna capacità

Nel caso in cui non vi sia un limite massimo C_i alla domanda che il centro i può soddisfare, il modello precedente può essere modificato in diverse maniere.

Una prima possibilità è procedere come già visto per i problemi di lot-sizing: manteniamo lo stesso modello, scegliendo per C_i dei valori sufficientemente grandi. Siccome in una soluzione ottima uno stesso centro soddisferà al massimo una domanda pari alla domanda totale dei clienti, possiamo prendere $C_i = \sum_{j=1}^{n} d_j$ per ogni i. In questo modo il limite superiore imposto dai vincoli (2.16) quando $y_i = 1$ è del tutto ininfluente.

Una seconda possibilità consiste nel rimpiazzare i vincoli (2.16) con le disequazioni

$$\sum_{j=1}^{n} x_{ij} \le n y_i, \quad i = 1,\dots,m.$$

Fissato i, questi vincoli fanno sì che se $x_{ij} > 0$ per qualche j (cioè il centro i viene utilizzato) allora $y_i = 1$. D'altro canto, quando $y_i = 1$ la disuguaglianza diventa superflua, dato che le x_{ij} valgono al massimo 1 e dunque la somma al primo membro non eccede n.

Un'ulteriore possibilità è quella di sostituire i vincoli (2.16) con le disequazioni

$$x_{ij} \leq y_i, \quad i = 1, \ldots, m,\ j = 1, \ldots, n.$$

L'effetto è identico a quello delle disequazioni precedenti.

Sebbene tutte queste possibilità portino a modelli equivalenti, dal punto di vista pratico può accadere che una scelta si riveli migliore di un'altra, a seconda della strategia utilizzata per risolvere il problema.

Variante: costi fissi sui trasporti

Supponiamo che vi sia anche, per ogni i e j, un costo fisso g_{ij} da pagare se viene effettuato un trasporto di merce da i a j (in altre parole, se $x_{ij} > 0$). Dovremo allora introdurre variabili binarie z_{ij} che valgono 1 se e solo se il trasporto da i a j viene attivato. La funzione obiettivo avrà il termine aggiuntivo $\sum_{i=1}^{m} \sum_{j=1}^{n} g_{ij} z_{ij}$. Inoltre scriveremo i vincoli

$$x_{ij} \leq z_{ij}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n.$$

Come al solito, la situazione $x_{ij} = 0$ e $z_{ij} = 1$, seppure ammissibile, non può verificarsi in una soluzione ottima.

Un'ultima variante è quella in cui è richiesto un costo fisso g_{ij} ogni volta che si trasporta un lotto di grandezza (al massimo) L_{ij} dal centro i al cliente j. In questo caso le variabili z_{ij} prenderanno valori interi e i vincoli scritti sopra saranno rimpiazzati dai seguenti:

$$d_j x_{ij} \leq L_{ij} z_{ij}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n.$$

2.2.6 Set covering, set packing e set partitioning

Illustriamo ora tre problemi (molto simili l'uno all'altro) la cui natura è più astratta di quella dei problemi finora incontrati, ma vedremo anche alcune possibili applicazioni.

Set covering

Sia I un insieme finito di cardinalità m e consideriamo una famiglia di sottoinsiemi di I: $S_1, \ldots, S_n \subseteq I$. Si chiede di selezionare il minor numero possibile di sottoinsiemi S_j in modo tale che la loro unione ricopra I. In altre parole, si deve trovare $J \subseteq \{1, \ldots, n\}$ in modo tale che $\bigcup_{j \in J} S_j = I$ e $|J|$ sia il più piccolo possibile.

Assumiamo senza perdita di generalità che $I = \{1, \ldots, m\}$. Conviene definire una matrice $A \in \{0, 1\}^{m \times n}$, le cui righe corrispondono agli elementi di I e le cui colonne corrispondono ai sottoinsiemi $S_1, \ldots, S_n$. Per ogni i e j, la componente a_{ij} di A è così definita:

$$a_{ij} = \begin{cases} 1 & \text{se } i \in S_j \\ 0 & \text{altrimenti.} \end{cases}$$

A è detta la *matrice di incidenza* della famiglia di sottoinsiemi $S_1, \ldots, S_n$ rispetto all'insieme I.

Per ogni $j = 1, \ldots, n$, definiamo una variabile binaria x_j che vale 1 se e solo se il sottoinsieme S_j viene scelto. L'i-esima componente del vettore Ax, cioè il numero $\sum_{j=1}^{n} a_{ij}x_j$, indicherà allora quanti dei sottoinsiemi scelti contengono l'elemento i: infatti il termine $a_{ij}x_j$ vale 1 se e solo se $a_{ij} = 1$ (cioè $i \in S_j$) e $x_j = 1$ (cioè S_j è stato scelto). Possiamo allora formulare il problema tramite il seguente programma lineare intero:

$$\begin{aligned} \min\ & \mathbf{1}^\top x \\ \text{s.a}\ & Ax \geq \mathbf{1} \\ & x \in \{0,1\}^n, \end{aligned}$$

dove $\mathbf{1}$ indica il vettore (di dimensione appropriata) con tutte le componenti uguali a 1. Si noti che $\mathbf{1}^\top x$ non è altro che una notazione abbreviata per l'espressione $\sum_{j=1}^{n} x_j$, che conta il numero di sottoinsiemi scelti. Il sistema $Ax \geq \mathbf{1}$ può anche essere scritto più esplicitamente:

$$\sum_{j:i \in S_j} x_j \geq 1, \quad i = 1, \ldots, m.$$

Possiamo immediatamente estendere la formulazione al caso in cui selezionare il sottoinsieme S_j abbia un costo c_j e siamo interessati a minimizzare il costo totale dei sottoinsiemi scelti: basterà sostituire la funzione obiettivo con $c^\top x$ (si parla di versione *pesata* del problema). Se poi per ogni i fosse richiesto che l'elemento i sia coperto da almeno b_i sottoinsiemi, dovremmo cambiare i vincoli in $Ax \geq b$.

Applicazione: Crew scheduling Consideriamo il problema di pianificare i turni del personale di bordo di una compagnia aerea, di cui sono noti tutti i voli effettuati. Gli equipaggi disponibili (ciascuno composto da pilota, copilota ed assistenti di volo) sono già stati formati e la loro composizione non può essere mutata. Anche i turni possibili sono già stati determinati in modo da soddisfare le norme sull'orario massimo di lavoro dei piloti. Ciascuno dei turni possibili è formato da una sequenza di voli (eventualmente uno solo) su cui l'equipaggio si imbarcherà. Come usuale nelle compagnie aeree, è anche possibile che su uno stesso volo vi sia più di un equipaggio, dei quali uno effettivamente in servizio e l'altro caricato allo scopo di raggiungere l'aeroporto di partenza del loro prossimo volo (in questo caso gli equipaggi che sono a bordo come passeggeri stanno comunque svolgendo il loro turno di lavoro). L'obiettivo è decidere quali dei turni possibili saranno attivati, in modo tale che tutti i voli siano dotati di (almeno) un equipaggio ed il numero di turni attivati (cioè di equipaggi in servizio) sia minimizzato.

Mostriamo come questo non sia altro che un problema di set covering. Definiamo I come l'insieme dei voli effettuati dalla compagnia. Poiché ognuno dei turni possibili è di fatto una sequenza di voli, possiamo definire formalmente un turno come un sottoinsieme di I. È ora chiaro che il problema consiste nel selezionare il minimo numero di sottoinsiemi (turni) la cui unione copra tutti gli elementi di I (i voli).

Set packing

Come sopra, abbiamo un insieme finito I di cardinalità m ed una famiglia di sottoinsiemi $S_1, \ldots, S_n \subseteq I$. Si chiede di selezionare il maggior numero possibile di sottoinsiemi S_j in modo tale che essi siano a due a due disgiunti.

Se chiamiamo A la matrice di incidenza e definiamo, per ogni j, una variabile binaria x_j che vale 1 se e solo se il sottoinsieme S_j viene scelto, il problema può essere formulato così:

$$\begin{aligned} \max\ & \mathbf{1}^\top x \\ \text{s.a}\ & Ax \leq \mathbf{1} \\ & x \in \{0,1\}^n. \end{aligned}$$

I vincoli assicurano che ogni elemento sia coperto da al massimo uno dei sottoinsiemi selezionati. Anche in questo caso l'estensione alla versione pesata è immediata.

Applicazione: Tandem linguistico Si consideri un'agenzia che mette in contatto persone interessate al tandem linguistico: ogni partecipante segnala all'agenzia la lingua straniera che vorrebbe apprendere e l'agenzia mette in contatto coppie di persone interessate l'una alla madrelingua dell'altra, affinché possano insegnarsi reciprocamente la lingua. L'agenzia dispone di una lista di nominativi (l'insieme I), a ciascuno dei quali è associata la lingua madre M_i e la lingua $L_i \neq M_i$ che la persona vorrebbe apprendere. Gli insiemi S_j descrivono gli accoppiamenti possibili: ogni S_j rappresenta una coppia di persone interessate l'una alla madrelingua dell'altra. Più formalmente, un sottoinsieme $S \subseteq I$ farà parte della famiglia $S_1, \ldots, S_n$ se e solo se $S = \{i_1, i_2\}$ per qualche $i_1, i_2 \in I$ tali che $L_{i_1} = M_{i_2}$ e $L_{i_2} = M_{i_1}$. Il problema di soddisfare il maggior numero possibile di clienti è un problema di set packing.[2]

Set partitioning

La situazione è la stessa, ma questa volta si chiede di selezionare il maggior o minor numero possibile di sottoinsiemi S_j in modo tale che essi formino una partizione di I. In altre parole, dobbiamo soddisfare contemporaneamente i vincoli di set packing e quelli di set covering. Dunque, con la solita notazione, scriveremo

$$\begin{aligned} \max\ (\text{o min})\ & \mathbf{1}^\top x \\ \text{s.a}\ & Ax = \mathbf{1} \\ & x \in \{0,1\}^n. \end{aligned}$$

Applicazioni Se nel problema di crew scheduling visto sopra non accettiamo la possibilità di imbarcare equipaggi come passeggeri (ad esempio perché non

[2] Per chi ha conoscenze di teoria dei grafi, questo è il caso particolare del problema del *matching massimo*.

vogliamo ridurre il numero di posti a disposizione dei clienti), otteniamo un problema di set partitioning.

Un altro esempio è dato dal problema di assegnamento illustrato nella Sezione 2.2.2: se definiamo I come l'insieme dei processi e dei processori, e come sottoinsiemi prendiamo tutti quelli contenenti esattamente un processo ed un processore, otteniamo un problema di set partitioning pesato.

2.2.7 Vincoli logici

Come ormai chiarito dai vari esempi, le variabili binarie vengono spesso utilizzate per descrivere decisioni del tipo sì/no. In altri termini, possiamo immaginare che ogni variabile binaria x_i corrisponda ad una proposizione logica X_i che è vera se e solo se $x_i = 1$. Con riferimento ai modelli precedenti, queste proposizioni X_i potrebbero essere, ad esempio, "Carico un certo oggetto nello zaino", "Assegno un certo lavoro ad una certa macchina", "Attivo la produzione in un certo periodo", "Seleziono un certo sottoinsieme".

Siano $x_1, \ldots, x_n$ le variabile binarie del nostro modello e indichiamo con $X_1, \ldots, X_n$ le corrispondenti proposizioni logiche, che chiamiamo variabili logiche. È possibilire modellizzare vincoli che descrivano proposizioni logiche più complesse, ottenute a partire dalle variabili logiche $X_1, \ldots, X_n$ tramite gli operatori logici di congiunzione, disgiunzione e negazione (indicati rispettivamente con $\wedge$, $\vee$, $\overline{\ }$)?

Sfruttando le proprietà distributive, la legge della doppia negazione e le leggi di De Morgan, è possibile scrivere qualunque proposizione logica in *forma normale congiuntiva*, cioè nella forma $P_1 \wedge \cdots \wedge P_m$, in cui ciascuna proposizione P_i è la disgiunzione di variabili logiche e/o variabili logiche negate (le P_i sono dette *clausole*). Un esempio di proposizione in forma normale congiuntiva (con quattro variabili logiche e tre clausole) è

$$(X_1 \vee \overline{X}_3) \wedge (\overline{X}_1 \vee X_2 \vee X_4) \wedge (\overline{X}_2 \vee \overline{X}_3 \vee \overline{X}_4). \tag{2.18}$$

Poiché la proposizione $P_1 \wedge \cdots \wedge P_m$ è vera se e solo se sono vere tutte le clausole che la compongono, basterà concentrarsi sulle singole clausole separatamente. Nell'esempio, la prima clausola $(X_1 \vee \overline{X}_3)$ è vera se e solo se almeno una tra X_1 e $\overline{X}_3$ è vera. Siccome X_1 è rappresentata dalla variabile x_1 e $\overline{X}_3$ dalla variabile complementata $1 - x_3$, la clausola $(X_1 \vee \overline{X}_3)$ è vera se e solo se $x_1 + (1 - x_3) \geq 1$. Procedendo in modo analogo per le altre clausole, possiamo descrivere la proposizione (2.18) tramite il seguente sistema di disequazioni nelle variabili binarie x_1, x_2, x_3, x_4:

$$\begin{aligned} &x_1 + (1 - x_3) \geq 1 \\ &(1 - x_1) + x_2 + x_4 \geq 1 \\ &(1 - x_2) + (1 - x_3) + (1 - x_4) \geq 1. \end{aligned}$$

L'esempio dovrebbe chiarire come procedere in generale: per ogni clausola si scrive una disequazione in cui il primo membro è la somma di variabili e/o variabili complementate, con secondo membro sempre uguale a 1.

Nel caso in cui si debba modellizzare un'implicazione logica, è sufficiente ricordare che $A \implies B$ è equivalente a $\overline{A} \vee B$. Per un'equivalenza logica, basta considerare le due implicazioni separatamente.

2.2.8 Disgiunzione di sistemi lineari

Siano dati m sistemi lineari $A^i x \leq b^i$ per $i = 1, \ldots, m$, tutti nelle variabili $x_1, \ldots, x_n$, e sia k un intero tale che $1 \leq k < m$. Supponiamo di avere un modello in cui vorremmo imporre la condizione che x deve soddisfare almeno k degli m sistemi lineari (oltre, eventualmente, ad altri vincoli che devono essere soddisfatti in ogni caso e che indichiamo con un sistema $Cx \leq d$). Mostriamo come ciò sia possibile tramite l'introduzione di variabili binarie.

Assumiamo che esita un numero M tale che ciascuno dei sistemi $A^i x \leq b^i + M\mathbf{1}$ sia ridondante per il modello (cioè sempre soddisfatto da qualunque x tale che $Cx \leq d$).[3] Per ogni $i = 1, \ldots, m$, introduciamo una variabile binaria y_i che vale 1 se e solo se il sistema $A^i x \leq b^i$ viene imposto. Possiamo allora scrivere la nostra regione ammissibile così:

$$Cx \leq d \tag{2.19}$$

$$A^i x \leq b^i + M\mathbf{1}(1 - y_i), \quad i = 1, \ldots, m \tag{2.20}$$

$$\sum_{i=1}^{m} y_i \geq k \tag{2.21}$$

$$y_i \in \{0, 1\}, \qquad i = 1, \ldots, m. \tag{2.22}$$

Si noti che quando $y_i = 0$ l'i-esimo sistema viene "disattivato". Il vinvolo (2.21) assicura che almeno k sistemi vengano "attivati".

Chiaramente potremmo anche imporre la condizione che al massimo o esattamente k sistemi siano soddisfatti semplicemente modificando la disuguaglianza (2.21). Anche l'estensione a sistemi contenenti disuguaglianze del tipo "$\geq$"è naturale. Qualora, invece, nei sistemi vi fossero equazioni, sarebbe necessario sostituire preventivamente ogni equazione con una coppia di disequazioni l'una l'opposta dell'altra.

Si noti infine come questa tecnica permetta di modellizzare vincoli del tipo "$\neq$", *purché le variabili che appaiono nel vincolo siano intere*: ad esempio, se le variabili x_1, x_2 sono intere, imporre il vincolo $2x_1 - 5x_2 \neq 7$ equivale a richiedere che x_1, x_2 soddisifino $2x_1 - 5x_2 \leq 6$ oppure $2x_1 - 5x_2 \geq 8$, in quanto il primo membro è necessariamente intero. Otteniamo dunque due disequazioni, delle quali vogliamo che una sia soddisfatta.

2.2.9 Linearizzazione

Talvolta si arriva a scrivere un problema di ottimizzazione che si avvicina molto all'essere un programma lineare intero (misto), ma in cui compaiono termini quadratici o di ordine superiore. Mostriamo qui come in certi casi sia possibile linearizzare questi termini, ottenendo così un programma lineare intero (misto).

[3] In problemi pratici un tale valore normalmente esiste ed è facile da identificare.

La trattazione che segue non considera tutte le possibilità immaginabili, ma dà comunque l'idea di come procedere (fermo restando che non tutti i modelli non-lineari sono linearizzabili).

Prodotto di variabili binarie

Come primo caso, supponiamo che nel modello siano presenti termini del tipo $x_1 x_2 \cdots x_k$ per un qualche $k > 1$, dove le variabili $x_1, \ldots, x_k$ sono tutte binarie. Definiamo $y = x_1 x_2 \cdots x_k$. Chiaramente y vale 1 se e solo se $x_i = 1$ per ogni $i = 1, \ldots, k$ e 0 altrimenti. L'idea è quella di sostituire ogni occorrenza di $x_1 x_2 \cdots x_k$ nel modello con la variabile binaria y, ma dobbiamo scrivere delle disuguaglianze lineari che impongano la condizione "$y = 1$ se e solo se $x_i = 1$ per ogni $i = 1, \ldots, k$". Questo può essere fatto, ad esempio, utilizzando quanto visto più sopra a proposito dei vincoli logici. Si ottengono le disuguaglianze

$$y \leq x_i, \quad i = 1, \ldots, k$$
$$y \geq x_1 + \cdots + x_k - (k-1).$$

Nel caso in cui $x_1 x_2 \cdots x_k$ non sia l'unico prodotto di variabili binarie presente nel modello, ad ogni prodotto diverso dovrà essere associata una diversa variabile binaria.

Prodotto tra variabile non-negativa e variabile binaria

Supponiamo ora di avere termini del tipo $x_1 x_2$, dove x_1 è una variabile non-negativa (continua o intera) e x_2 è una variabile binaria. Supponiamo di conoscere un valore M tale che $x_1 \leq M$ per tutte le soluzioni ammissibili ($M \geq 0$ senza perdita di generalità). Sostituiamo ogni occorrenza di $x_1 x_2$ con una nuova variabile y e scriviamo i vincoli

$$0 \leq y \leq M x_2$$
$$x_1 - M(1 - x_2) \leq y \leq x_1.$$

Quando $x_2 = 0$, il primo vincolo diventa $y = 0$; e con $x_2 = y = 0$, il secondo vincolo dà le disequazioni $x_1 - M \leq 0 \leq x_1$, che sono sempre soddisfatte. Quando invece $x_2 = 1$, il secondo vincolo diventa $y = x_1$; e con $y = x_1$, il primo vincolo dà le disequazioni $0 \leq x_1 \leq M$, che sono sempre soddisfatte. Dunque i vincoli scritti sopra sono equivalenti a

$$y = \begin{cases} x_1 & \text{se } x_2 = 1 \\ 0 & \text{se } x_2 = 0, \end{cases}$$

cioè $y = x_1 x_2$.

Prodotto tra variabile non-negativa e più variabili binarie

Le due trasformazioni viste sopra possono essere usate anche in combinazione. Ad esempio, nel caso ci sia un termine del tipo $x_1 x_2 \cdots x_k$, dove $0 \leq x_1 \leq M$ e $x_2, \ldots, x_k \in \{0, 1\}$, si può procedere in due fasi: prima sostituiamo il prodotto $x_2 \cdots x_k$ con una variabile binaria y (aggiungendo le dovute disuguaglianze viste sopra) e poi trattiamo il prodotto $x_1 y$ come nel caso precedente.

2.2.10 Variabili con dominio finito

Supponiamo di avere un modello in cui vogliamo imporre la condizione che una certa variabile x prenda un valore in un dato insieme finito $\{a_1, \ldots, a_m\} \subseteq \mathbb{R}$. Introduciamo variabili binarie $y_1, \ldots, y_m$, dove, per ogni $i = 1, \ldots, m$, y_i vale 1 se e solo se x prende il valore a_i. Sarà allora sufficiente includere nel modello i vincoli

$$x = \sum_{i=1}^{m} a_i y_i \tag{2.23}$$

$$\sum_{i=1}^{m} y_i = 1 \tag{2.24}$$

$$y_i \in \{0, 1\}, \quad i = 1, \ldots, m, \tag{2.25}$$

dove l'equazione (2.24) e i vincoli (2.25) fanno sì che una e una sola delle variabili $y_1, \ldots, y_m$ valga 1 (e le altre 0).

2.3 Branch-and-bound

I problemi di programmazione lineare intera ed intera mista sono, di norma, molto più difficili da risolvere rispetto a quelli di programmazione lineare. Per questo motivo sono state sviluppate varie tecniche risolutive, nessuna delle quali però mostra una supremazia assoluta sulle altre: a seconda del problema che ci si trova di fronte, una strategia può risultare migliore di un'altra. Inoltre, è spesso necessario sfruttare le caratteristiche specifiche del particolare problema che si vuole risolvere.

Quella illustrata in questa sezione è una tecnica di carattere generale molto usata nella pratica, detta *branch-and-bound*. Per introdurla, è necessario definire prima il rilassamento lineare di un programma lineare intero (misto).

2.3.1 Il rilassamento lineare

Un primo naturale tentativo per risolvere programmi lineari interi (misti) è quello di considerare il *rilassamento lineare* (o rilassamento continuo) del problema. Il rilassamento lineare si ottiene semplicemente tralasciando i vincoli di interezza $x_i \in \mathbb{Z}$ (nel caso in cui una variabile sia binaria, il vincolo $x_i \in \{0, 1\}$ viene sostituito dalle disuguaglianze $0 \leq x_i \leq 1$). Il problema così ottenuto è un programma lineare e può quindi essere risolto in modo efficiente. Tuttavia già con sole due variabili il valore ottimo del rilassamento lineare può essere arbitrariamente distante dal valore ottimo del programma lineare intero di partenza. Ad esempio, si consideri il programma lineare intero

$$\begin{aligned} \max \quad & x_2 \\ \text{s.a} \quad & kx_1 - x_2 \geq 0 \\ & kx_1 + x_2 \leq k \\ & x_2 \geq 0 \\ & x_1, x_2 \in \mathbb{Z}, \end{aligned}$$

dove k è un qualunque numero positivo. Ignorando i vincoli di interezza, la regione ammissibile è il triangolo con vertici $(0,0)$, $(1,0)$ e $(1/2, k/2)$, dunque il valore ottimo del rilassamento lineare è $k/2$. Considerando invece anche i vincoli di interezza, la regione ammissibile contiene soltanto i punti $(0,0)$ e $(1,0)$, dunque il valore ottimo del programma lineare intero è 0. La differenza tra i due valori ottimi è quindi pari a $k/2$, che può essere grande a piacere.

Dunque il rilassamento lineare non fornisce, in generale, una buona approssimazione del problema. Ciononostante, vedremo che vari metodi risolutivi per la programmazione lineare intera (mista) fanno ripetutamente uso del rilassamento lineare.

2.3.2 L'algoritmo di branch-and-bound

In quanto segue assumiamo che il programma lineare intero assegnato sia un problema di massimizzazione. In caso contrario, si dovrà avere cura di scambiare tra loro "upper bound" e "lower bound", oltre ovviamente a massimi e minimi. Assumiamo inoltre, almeno per il momento, che il problema sia di programmazione lineare intera pura, cioè che non vi siano variabili continue.

Il metodo di branch-and-bound è una tecnica di enumerazione implicita delle soluzioni ammissibili. Per "enumerazione implicita" si intende che nel cercare una soluzione ottima non si esaminano effettivamente tutte le soluzioni ammissibili, perché, come vedremo, una parte di esse potrà essere scartata.

Il branch-and-bound si basa essenzialmente su due osservazioni elementari, alle quali l'algoritmo deve il suo nome (la parte di *bounding* sfrutta il primo dei due lemmi, quella di *branching* il secondo).

Lemma 2.1 *Il valore ottimo del rilassamento lineare è un upper bound al valore ottimo del programma lineare intero.*

Dimostrazione. Basta osservare che ogni soluzione ammissibile del programma lineare intero è ammissibile anche per il rilassamento lineare. □

Lemma 2.2 *Se la regione ammissibile Ω di un problema di ottimizzazione viene suddivisa in due sottoinsiemi (diciamo $\Omega = \Omega_1 \cup \Omega_2$), il valore ottimo del problema iniziale è il migliore dei valori ottimi dei due sottoproblemi:*

$$\max\{c^\top x : x \in \Omega\} = \max\big\{\max\{c^\top x : x \in \Omega_1\}, \max\{c^\top x : x \in \Omega_2\}\big\}.$$

Dimostrazione. Lampante. □

Si noti che vale anche il seguente facile risultato.

Osservazione 2.3 *Se la (o una) soluzione ottima del rilassamento lineare è intera, allora tale soluzione è ottima anche per il programma lineare intero. Se il rilassamento lineare è inammissibile, allora anche il programma lineare intero è inammissibile.*

L'algoritmo di branch-and-bound è una tecnica ricorsiva. Inizialmente si risolve il rilassamento lineare del programma lineare intero assegnato. Per l'Osservazione 2.3, se la soluzione ottima del rilassamento lineare è intera o se non ci sono soluzioni ammissibili, allora abbiamo terminato. Altrimenti, la soluzione ottima $\bar{x}$ del rilassamento lineare ha almeno una componente $\bar{x}_i \notin \mathbb{Z}$. Consideriamo allora i due sottoproblemi ottenuti dal programma lineare intero di partenza aggiungendo in un caso il vincolo $x_i \leq \lfloor \bar{x}_i \rfloor$, nell'altro caso il vincolo $x_i \geq \lceil \bar{x}_i \rceil$. Questa operazione è detta *branching*. Poiché ogni soluzione ammissibile intera del problema iniziale soddisfa i vincoli di uno dei due sottoproblemi, il Lemma 2.2 implica che risolvendo questi due sottoproblemi e scegliendo la migliore delle due soluzioni trovate otterremo una soluzione ottima del programma lineare intero di partenza. A questo punto si itera: per ciascuno dei due sottoproblemi, si risolve il corrispondente rilassamento lineare, e così via. Si noti che in nessuno dei due rilassamenti lineari potremo trovare la stessa soluzione $\bar{x}$ trovata per il problema padre, perché $\bar{x}$ non è ammissibile per nessuno dei due sottoproblemi (nemmeno per i loro rilassamenti lineari).

Questo procedimento ricorsivo fa raddoppiare ad ogni iterazione il numero di sottoproblemi in esame. Ma è davvero necessario esaminare tutti i sottoproblemi creati? La risposta è no: ci sono tre criteri che possono essere usati per stabilire che un certo sottoproblema non deve essere esaminato ulteriormente (si parla di *potatura* del sottoproblema).

(a) Se il rilassamento lineare di un sottoproblema P è inammissibile, allora per l'Osservazione 2.3 anche P stesso è inammissibile: possiamo potare per inammissibilità.

(b) Se il rilassamento lineare di un sottoproblema P ha una soluzione ottima $\bar{x}$ che è un vettore intero, per l'Osservazione 2.3 tale soluzione è ottima anche per P. In questo caso procediamo come segue:

- Se $\bar{x}$ è la prima soluzione ammissibile intera che abbiamo trovato durante l'algoritmo, la registriamo come soluzione *incombente* (è questo il nome dato alla migliore soluzione intera finora trovata).
- Se invece una soluzione intera $\tilde{x}$ era già stata trovata in qualche altro sottoproblema P' ($\tilde{x}$ è l'incombente corrente), confrontiamo le due soluzioni $\bar{x}$ e $\tilde{x}$: se $\bar{x}$ è migliore di $\tilde{x}$, registriamo $\bar{x}$ come nuova soluzione incombente, potiamo P per ottimalità (non possiamo trovare soluzioni migliori in P) e potiamo P' per bound (dato che l'ottimo di P' viene battuto da $\bar{x}$, non ha senso cercare altre soluzioni in P'); se invece $\bar{x}$ non è migliore di $\tilde{x}$, potiamo P per bound senza aggiornare l'incombente.

(c) Se il valore ottimo del rilassamento lineare di un sottoproblema P è peggiore dell'incombente (o al massimo uguale), il Lemma 2.1 ci permette di potare P per bound, anche se non conosciamo il valore ottimo di P (abbiamo solo quello del rilassamento lineare).

L'algoritmo termina quando tutti i sottoproblemi sono stati potati.

Libertà

Durante l'esecuzione del branch-and-bound, ci si trova spesso a dover effettuare delle scelte che possono influenzare pesantemente il numero di iterazioni necessarie al completamento dell'algoritmo. Le principali fasi in cui c'è libertà di azione sono elencate qui sotto. Purtroppo non esistono regole precise che stabiliscano quando effettuare una scelta piuttosto che un'altra.

- Consideriamo un qualche sottoproblema P. Dopo aver effettuato il branching (ottenendo i sottoproblemi P' e P'') ed aver risolto il rilassamento lineare di P', risolviamo P'' o eseguiamo prima il branching su P'? La prima scelta è la strategia del *breadth first*, la seconda quella del *depth first*.

- Quando dobbiamo effettuare un branching e ci sono più variabili con valore non intero, quale scegliamo?

- Una volta effettuato il branching, quale dei due sottoproblemi risolviamo per primo?

Nei principali risolutori in commercio, queste scelte (così come molte altre) possono essere fatte dall'utente oppure lasciate al software.

Simplesso duale

Nell'applicazione del branch-and-bound, il rilassamento lineare del problema di partenza è generalmente quello che richiede lo sforzo maggiore per essere risolto. Ciascuno dei problemi successivi può infatti essere risolto rapidamente grazie all'algoritmo del simplesso duale, come ora illustriamo.[4]

Si ricordi che l'algoritmo del simplesso restituisce, oltre ad una soluzione ottima del primale, anche una soluzione ottima del duale. In quanto segue, quando si parla di primale e duale di un problema si intendono sempre il primale ed il duale del rilassamento lineare del problema.

Come già discusso in precedenza, l'operazione di branching genera due sottoproblemi P' e P'', ciascuno dei quali contiene un vincolo in più rispetto al problema padre P; inoltre, tale vincolo è violato dalla soluzione ottima del rilassamento lineare di P. Tuttavia, è facile verificare che una soluzione ottima del duale di P si estende immediatamente ad una soluzione ammissibile dei duali di P' e P''. Questa soluzione può essere usata come soluzione di base di partenza per l'esecuzione del metodo del simplesso sui duali di P' e P''. L'esperienza mostra che, partendo da tale soluzione ammissibile, si arriva molto rapidamente alle soluzioni ottime dei duali di P' e P'' (e dunque anche a quelle dei primali). Se invece si volesse applicare il metodo del simplesso direttamente ai problemi primali P', P'', non disponendo di una soluzione di base da cui partire, l'esecuzione risulterebbe molto più lunga (intuitivamente, si dovrebbe ripetere lo sforzo già effettuato per risolvere i sottoproblemi precedenti).

[4] Questo non significa che una volta risolto il primo rilassamento lineare siamo vicini alla conclusione dell'algoritmo, perché il numero di sottoproblemi potrebbe essere enorme.

Stima dell'errore

In molti problemi di interesse pratico, è impossibile eseguire il metodo del branch-and-bound fino alla sua naturale terminazione, in quanto lo sforzo computazionale richiesto può essere smisurato anche per i più moderni calcolatori. Nel caso si debba interrompere l'algoritmo anticipatamente e ci si debba accontentare di una soluzione intera che potrebbe non essere ottima, è utile avere la garanzia che tale soluzione intera non disti troppo dall'ottimo.

Considerando, come sopra, un problema di massimizzazione, sia z^* il valore ottimo del programma lineare intero di partenza (valore che dunque non conosciamo). Quando interrompiamo l'algoritmo di branch-and-bound, abbiamo a disposizione un lower bound z^- ed un upper bound z^+ sul valore ottimo: il lower bound è dato dall'incombente, mentre il miglior upper bound può essere trovato cercando tra le soluzioni ottime dei rilassamenti lineari dei sottoproblemi che non sono ancora stati potati: è infatti in questi sottoproblemi che potrebbe celarsi una soluzione migliore di quella fin qui trovata. Assumiamo per semplicità che i valori z^*, z^-, z^+ siano tutti positivi. Allora l'errore assoluto che commettiamo nell'accettare l'incombente come soluzione è

$$E_{\mathrm{A}} = z^* - z^- \leq z^+ - z^-,$$

mentre l'errore relativo è

$$E_{\mathrm{R}} = \frac{z^* - z^-}{z^*} \leq \frac{z^+ - z^-}{z^-}.$$

In entrambi i casi il valore dell'ultimo membro è noto, dunque abbiamo una garanzia sul massimo errore che stiamo commettendo.

Il caso della programmazione lineare intera mista

Nel caso in cui il problema da risolvere sia un programma lineare intero misto, la discussione fin qui svolta resta valida, con la sola eccezione che, nell'effettuare il branching, considereremo soltanto quelle variabili per cui il problema richiede l'interezza.

2.4 Approccio poliedrale

Un secondo approccio risolutivo, anche questo molto utilizzato nella pratica, sfrutta gli aspetti poliedrali della programmazione lineare intera (mista).

2.4.1 Inviluppo convesso della regione ammissibile

Un *poliedro* è l'insieme dei punti in $\mathbb{R}^n$ che risolvono un dato sistema di equazioni e/o disequazioni lineari. In altre parole, un poliedro è l'intersezione di un numero finito di semispazi chiusi. Si noti che la regione ammissibile di un qualunque programma lineare è un poliedro, così come la regione ammissibile del rilassamento lineare di un qualunque programma lineare intero (misto).

Dati k punti $x^1, \ldots, x^k \in \mathbb{R}^n$, un punto $x \in \mathbb{R}^n$ è una *combinazione convessa* di $x^1, \ldots, x^k$ se può essere scritto nella forma

$$x = \sum_{i=1}^{k} \lambda_i x^i, \text{ dove } \sum_{i=1}^{k} \lambda_i = 1 \text{ e } \lambda_1, \ldots, \lambda_k \geq 0. \tag{2.26}$$

Dato un poliedro $P \subseteq \mathbb{R}^n$, un punto $x \in P$ è un *vertice* (o *punto estremo*) di P se x non è combinazione convessa di punti di P diversi da x. Dato un programma lineare in forma standard, è possibile dimostrare che le soluzioni ammissibili di base sono esattamente i vertici della regione ammissibile. In particolare, la soluzione ottima restituita dal metodo del simplesso è un vertice della regione ammissibile.

Dato un qualunque insieme $\Omega \subseteq \mathbb{R}^n$, l'inviluppo convesso di Ω, indicato con $\text{conv}(\Omega)$, è l'insieme di tutte le combinazioni convesse di punti in Ω. È facile verificare che $\text{conv}(\Omega)$ è il più piccolo insieme convesso contenente Ω, nonché l'intersezione di tutti gli insiemi convessi contenenti Ω.

Il risultato seguente mostra che per problemi di ottimizzazione con funzione obiettivo lineare, ottimizzare sulla regione ammissibile data o sul suo inviluppo convesso è essenzialmente equivalente.

Lemma 2.4 *Dati un insieme $\Omega \subseteq \mathbb{R}^n$ ed un vettore $c \in \mathbb{R}^n$, i valori ottimi dei due problemi seguenti coincidono:*

$$\max\{c^\top x : x \in \Omega\} = \max\{c^\top x : x \in \text{conv}(\Omega)\}.$$

Dimostrazione. Poiché la regione ammissibile del secondo problema contiene quella del primo, il secondo problema non può avere valore ottimo peggiore di quello del primo. Per mostrare la disuguaglianza inversa, fissiamo una qualunque soluzione ammissibile del secondo problema, cioè $x \in \text{conv}(\Omega)$, e mostriamo che il primo problema ha una soluzione ammissibile $\tilde{x} \in \Omega$ tale che $c^\top \tilde{x} \geq c^\top x$. Per definizione di inviluppo convesso, x è della forma (2.26), con $x^1, \ldots, x^k \in \Omega$. Ora sosteniamo che $c^\top x^i \geq c^\top x$ per almeno un indice i. Se così non fosse, avremmo

$$c^\top x = \sum_{i=1}^{k} \lambda_i c^\top x^i < \sum_{i=1}^{k} \lambda_i c^\top x = c^\top x,$$

dove la disuguaglianza vale perché i λ_i sono tutti non-negativi ed almeno uno di essi è positivo. Abbiamo ottenuto una contraddizione, dunque $c^\top x^i \geq c^\top x$ per almeno un indice i. Poiché $x^i \in \Omega$, possiamo scegliere $\tilde{x} = x^i$. □

Grazie al teorema che ora enunciamo, il lemma precedente diventa particolarmente interessante nel caso della programmazione lineare intera (mista).

Teorema 2.5 (Meyer 1974) *Sia Ω la regione ammissibile di un programma lineare intero (misto) con dati razionali. Allora* $\text{conv}(\Omega)$ *è un poliedro.*

Segnaliamo che l'ipotesi di razionalità dei dati può essere rimossa se Ω è un insieme limitato.

Consideriamo un programma lineare intero (misto)

$$\max\ c^\top x \tag{2.27}$$
$$\text{s.a } Ax \leq b \tag{2.28}$$
$$x_i \in \mathbb{Z}, \quad i \in I, \tag{2.29}$$

dove tutte le componenti di A e b sono numeri razionali, ed indichiamo con Ω la regione ammissibile (2.28)–(2.29). Il Teorema 2.5 garantisce che $\text{conv}(\Omega)$ è un poliedro ed è dunque descritto da un qualche sistema $Cx \leq d$. Per il Lemma 2.4, il problema (2.27)–(2.29) è allora equivalente al programma lineare

$$\max\ c^\top x \tag{2.30}$$
$$\text{s.a } Cx \leq d. \tag{2.31}$$

Questo significa che, in linea di principio, ogni programma lineare intero (misto) con dati razionali è equivalente ad un qualche programma lineare. Questa è un'informazione molto utile, perché i programmi lineari possono essere risolti efficacemente. Tuttavia, vi sono almeno due considerazioni da fare.

Il Lemma 2.4 garantisce che i *valori* ottimi dei due problemi (2.27)–(2.29) e (2.30)–(2.31) sono gli stessi, ma non che le *soluzioni* ottime coincidano. In effetti il secondo problema può avere più soluzioni ottime del primo. Potrebbe dunque accadere che risolvendo il problema (2.30)–(2.31) si trovi una soluzione ottima che non appartiene ad Ω, cioè che non soddisfa (2.28)–(2.29). Tuttavia, questa eventualità può essere esclusa nel caso in cui si utilizzi il metodo del simplesso per risolvere il secondo problema: in tal caso, infatti, troveremo una soluzione ottima (se ne esiste una) che è un vertice di $\text{conv}(\Omega)$ (cioè del poliedro definito dal sistema $Cx \leq d$). Ma i vertici di $\text{conv}(\Omega)$ sono necessariamente punti di Ω, come ora mostriamo.

Osservazione 2.6 *Sia $\Omega \subseteq \mathbb{R}^n$ e supponiamo che $\text{conv}(\Omega)$ sia un poliedro. Allora i vertici di $\text{conv}(\Omega)$ sono punti di Ω.*

Dimostrazione. Un vertice x di $\text{conv}(\Omega)$ non può essere espresso come combinazione convessa di punti di $\text{conv}(\Omega)$ diversi da x. D'altro canto, in quanto punto di $\text{conv}(\Omega)$, x è combinazione convessa di punti $x^1, \ldots, x^k \in \Omega$. Ne segue che ciascuno degli x^i coincide con x, dunque $x \in \Omega$. □

Una seconda, ben più problematica, considerazione è la seguente: la possibilità di convertire il programma lineare intero (misto) assegnato in un programma lineare è subordinata alla conoscenza di un sistema $Cx \leq d$ che descriva l'inviluppo convesso della regione data Ω. In generale, è purtroppo estremamente difficile determinare un tale sistema. Inoltre, anche partendo da un programma lineare intero (misto) descritto da pochissimi vincoli, è possibile che l'inviluppo convesso della regione ammissibile sia definito da un numero enorme di disequazioni. Dunque questa equivalenza teorica tra programmazione lineare intera (mista) e programmazione lineare non può essere applicata direttamente nella pratica.

2.4.2 Algoritmi di tipo poliedrale: piani di taglio

Considerata la difficoltà di trovare un sistema $Cx \leq d$ che descriva l'inviluppo convesso della regione ammissibile data, si può pensare ad un approccio più mirato, come quello presentato qui sotto dopo alcune definizioni.

Dato un insieme $\Omega \subseteq \mathbb{R}^n$, una disequazione $a^\top x \leq \beta$ è una *disuguaglianza valida* per Ω se è soddisfatta da tutti i punti di Ω. È immediato verificare che una disuguaglianza è valida per Ω se e solo se è valida per $\text{conv}(\Omega)$.

Dato un programma lineare intero (misto) con regione ammissibile Ω, una disequazione $a^\top x \leq \beta$ è un *piano di taglio* (a volte semplicemente *taglio*) se è valida per Ω (e dunque per $\text{conv}(\Omega)$) ma è violata da almeno un punto del rilassamento lineare del problema. Possiamo interpretare un piano di taglio come una disequazione che contribuisce a "tagliare via" dei punti che giacciono al di fuori dell'inviluppo convesso della regione ammissibile.

I piani di taglio possono essere sfruttati in un algoritmo iterativo che ora presentiamo. Per semplicità di descrizione, assumiamo di voler risolvere un programma lineare intero puro, ma l'estensione al caso intero misto è immediata. Come sopra, indichiamo con Ω la regione ammissibile del problema dato.

Inizialmente risolviamo il rilassamento lineare del programma lineare intero dato, trovando una soluzione ottima $\bar{x}$. Se $\bar{x}$ è un punto intero, abbiamo terminato. Altrimenti cerchiamo un piano di taglio $a^\top x \leq \beta$ che sia valido per Ω ma violato da $\bar{x}$ (cioè $a^\top \bar{x} > \beta$). Aggiungiamo la disequazione $a^\top x \leq \beta$ al rilassamento lineare del problema e ripetiamo la procedura. (Si noti che, poiché ad ogni iterazione viene aggiunto un singolo vincolo e tale vincolo è violato dalla soluzione ottima corrente, all'iterazione successiva si potrà riottimizzare rapidamente usando il metodo del simplesso duale, esattamente come osservato per il branch-and-bound nella Sezione 2.3.2).

Tale algoritmo cerca di raggiungere l'ottimo intero eliminando ad ogni passo la soluzione ottima del rilassamento lineare appena trovata. È chiaro che ad ogni iterazione ci si avvicinerà alla soluzione ottima intera (salvo iterazioni degeneri in cui si trova una soluzione diversa ma con lo stesso valore della funzione obiettivo). Tuttavia ci sono due questioni da trattare: (i) come trovare i piani di taglio necessari; (ii) stabilire se questa procedura converge alla soluzione ottima intera. Rispondiamo a queste due domande considerando separatamente il caso intero puro e quello intero misto. Sebbene esistano vari modi di generare tagli, ci concentreremo sulle tecniche più comuni.

2.4.3 Tagli di Gomory

Consideriamo un programma lineare intero puro con dati razionali. Moltiplicando tutte le disequazioni per un appropriato numero intero, possiamo assumere che tutti i coefficienti siano numeri interi. Procedendo come nel caso della programmazione lineare, il nostro problema può essere messo in forma

standard:

$$\max\ c^\top x \tag{2.32}$$
$$\text{s.a}\ Ax = b \tag{2.33}$$
$$x \geq \mathbf{0} \tag{2.34}$$
$$x \in \mathbb{Z}^n. \tag{2.35}$$

Alcune delle variabili x_i saranno ovviamente variabili scarto aggiunte al problema iniziale per passare alla forma standard. Il fatto che i coefficienti siano tutti interi permette di imporre la condizione di interezza anche su tali variabili, mentre in caso contrario le variabili scarto dovrebbero essere variabili continue. Questo è importante perché il procedimento qui presentato si applica solo ai programmi lineari interi puri.

Supponiamo di aver risolto il rilassamento lineare del problema col metodo del simplesso, ottenendo una soluzione ottima di base $\bar{x}$ associata ad una base B. Come visto nella Sezione 1.1.2, il sistema $Ax = b$ è equivalente al sistema

$$x_B = A_B^{-1}b - A_B^{-1}A_N x_N \tag{2.36}$$

e la soluzione $\bar{x}$ è definita da $\bar{x}_B = A_B^{-1}b$, $\bar{x}_N = \mathbf{0}$. Assumiamo che $\bar{x}$ non sia un vettore intero, altrimenti abbiamo terminato. Dunque esiste un indice j tale che $\bar{x}_j \notin \mathbb{Z}$ e chiaramente $j \in B$, dato che le componenti di $\bar{x}$ in non-base valgono zero. La riga del sistema (2.36) corrispondente alla variabile x_j avrà la forma

$$x_j = \beta - \sum_{i \in N} \alpha_i x_i \tag{2.37}$$

per opportuni coefficienti α_i e β. Si noti che $\bar{x}_j = \beta$, dunque $\beta \notin \mathbb{Z}$.

Proposizione 2.7 *La seguente disequazione è un piano di taglio per la regione ammissibile* (2.33)–(2.35)*, violato da* $\bar{x}$*:*

$$x_j + \sum_{i \in N} \lfloor \alpha_i \rfloor x_i \leq \lfloor \beta \rfloor . \tag{2.38}$$

Dimostrazione. Poiché tutte le soluzioni ammissibili x del programma lineare intero iniziale soddisfano (2.37) e $x \geq \mathbf{0}$, la seguente disequazione è valida per la regione ammissibile:

$$x_j + \sum_{i \in N} \lfloor \alpha_i \rfloor x_i \leq \beta. \tag{2.39}$$

Ora, sfruttando il fatto che $x \in \mathbb{Z}^n$ per tutte le soluzioni ammissibili ed osservando che tutti i coefficienti $\lfloor \alpha_i \rfloor$ sono interi, vediamo che il primo membro della disequazione (2.39) deve assumere un valore intero. Ne segue che la disequazione (2.38) è valida per la regione ammissibile (2.33)–(2.35).

Per concludere, dobbiamo mostrare che $\bar{x}$ non soddisfa (2.38). Ma questo segue semplicemente dal fatto che $\bar{x}_i = 0$ per $i \in N$ e $\bar{x}_j = \beta > \lfloor \beta \rfloor$. □

La disequazione (2.38) è un *taglio di Gomory*.

Quanto appena descritto permette di ricavare un taglio di Gomory in maniera automatica a partire dall'output del metodo del simplesso. Questa procedura può quindi essere usata all'interno di un algoritmo come quello descritto nella Sezione 2.4.2. La domanda a questo punto è: utilizzando i tagli di Gomory, l'algoritmo converge alla soluzione ottima intera? Come mostrato da Gomory nel 1958, è possibile fissare una regola che permette di avere la garanzia di raggiungere l'ottimo intero entro un numero finito di iterazioni. Tale regola, da applicare ad ogni iterazione, stabilisce essenzialmente quale soluzione ottima del rilassamento lineare considerare nel caso in cui ce ne sia più di una.

2.4.4 Tagli di Gomory interi misti

La dimostrazione della correttezza dei tagli di Gomory vista nella Sezione 2.4.3 consta essenzialmente di due passaggi: prima si indebolisce l'equazione (2.37) in modo da avere solo coefficienti interi a primo membro; poi si sfrutta il fatto che tutte le variabili del problema devono assumere valore intero. La seconda parte è un semplice argomento di arrotondamento: se un'espressione z deve avere valore intero e soddisfa la disuguaglianza $z \leq \beta$, allora deve necessariamente valere $z \leq \lfloor \beta \rfloor$.

Nel caso intero misto tale approccio non può funzionare, perché non tutte le variabili sono vincolate ad assumere valore intero. Tuttavia, è possibile sfruttare un argomento dello stesso tipo, anche se un po' più sofisticato. In questa sezione otterremo un piano di taglio per problemi di programmazione lineare intera mista facendo di nuovo uso di un procedimento in due tappe: indebolimento di un'equazione valida e, in seguito, arrotondamento. Il lemma che segue permetterà di effettuare l'arrotondamento in modo corretto.

Lemma 2.8 *Sia $\beta \in \mathbb{R} \setminus \mathbb{Z}$ e sia f la parte frazionaria di β, cioè $f = \beta - \lfloor \beta \rfloor > 0$. Se un vettore (s, z) soddisfa i vincoli*

$$z \leq \beta + s$$
$$s \geq 0, z \in \mathbb{Z},$$

allora soddisfa anche la disuguaglianza

$$z \leq \lfloor \beta \rfloor + \frac{s}{1-f}. \tag{2.40}$$

Dimostrazione. Se $z \leq \lfloor \beta \rfloor$ allora la (2.40) segue dal fatto che, siccome $f < 1$, si ha $\frac{s}{1-f} \geq 0$. Dunque supponiamo che $z \geq \lfloor \beta \rfloor + 1$. Poiché $z \leq \beta + s$ e $f < 1$, abbiamo

$$\frac{1}{1-f} z \leq \frac{\beta + s}{1-f};$$

e poiché $-z \leq -(\lfloor \beta \rfloor + 1)$, abbiamo

$$-\left(\frac{1}{1-f} - 1\right) z \leq -\left(\frac{1}{1-f} - 1\right)(\lfloor \beta \rfloor + 1).$$

Sommando le ultime due disuguaglianze e ricordando che $f = \beta - \lfloor \beta \rfloor$, otteniamo (2.40). □

La disequazione (2.40) è detta *MIR* (*mixed-integer rounding inequality*).

Consideriamo un programma lineare intero misto. Indicando con x_i le variabili intere e con y_i quelle continue ed assumendo senza perdita di generalità che il programma sia in forma standard, il problema è il seguente:

$$\max\ c^\top x \tag{2.41}$$
$$\text{s.a}\ Ax + By = d \tag{2.42}$$
$$x \geq \mathbf{0},\ y \geq \mathbf{0} \tag{2.43}$$
$$x \text{ intero.} \tag{2.44}$$

Supponiamo di aver risolto il rilassamento lineare del problema col metodo del simplesso, ottenendo una soluzione ottima di base $(\bar{x}, \bar{y})$ associata ad una base B e una non-base N. Scriviamo $N = N_1 \cup N_2$, dove N_1 contiene gli indici in N corrispondenti a variabili intere e N_2 contiene gli indici in N corrispondenti a variabili continue. Assumiamo che $\bar{x}$ non sia un vettore intero, altrimenti abbiamo terminato. Dunque esiste un indice $j \in B$ tale che $\bar{x}_j \notin \mathbb{Z}$. Come sempre, il metodo del simplesso ci restituisce un sistema equivalente a (2.42) in cui le variabili in base sono esplicitate. La riga di tale sistema corrispondente alla variabile x_j avrà la forma

$$x_j + \sum_{i \in N_1} \alpha_i x_i + \sum_{i \in N_2} \alpha_i y_i = \beta \tag{2.45}$$

per opportuni coefficienti α_i e β. Si noti che $\bar{x}_j = \beta$, dunque $\beta \notin \mathbb{Z}$. Indichiamo con $f_0 = \beta - \lfloor \beta \rfloor$ la parte frazionaria di β e con $f_i = \alpha_i - \lfloor \alpha_i \rfloor$ quella di α_i, per $i \in N$. Si noti che $f_0 > 0$. Infine, dato un numero reale a, indichiamo con a^+ e a^- rispettivamente la parte positiva e la parte negativa di a, cioè $a^+ = \max\{a, 0\}$ e $a^- = \max\{-a, 0\}$.

Proposizione 2.9 *La seguente disequazione è un piano di taglio per la regione ammissibile* (2.42)–(2.44)*, violato da* $(\bar{x}, \bar{y})$*:*

$$x_j + \sum_{i \in N_1} \left(\lfloor \alpha_i \rfloor + \frac{(f_i - f_0)^+}{1 - f_0} \right) x_i \leq \lfloor \beta \rfloor + \sum_{i \in N_2} \frac{\alpha_i^-}{1 - f_0} y_i. \tag{2.46}$$

Dimostrazione. Partendo dalla validità dell'equazione (2.45) e sfruttando il fatto che tutte le variabili sono non-negative, si osserva che la disequazione

$$x_j + \sum_{\substack{i \in N_1 \\ f_i \leq f_0}} \lfloor \alpha_i \rfloor\, x_i + \sum_{\substack{i \in N_1 \\ f_i > f_0}} \alpha_i x_i \leq \beta + \sum_{i \in N_2} \alpha_i^- y_i$$

è valida per la regione ammissibile. Osservando che $\alpha_i = (\lfloor \alpha \rfloor_i + 1) - (1 - f_i)$ per ogni i, possiamo decomporre la seconda sommatoria in due pezzi, ottenendo

$$x_j + \sum_{\substack{i \in N_1 \\ f_i \leq f_0}} \lfloor \alpha_i \rfloor\, x_i + \sum_{\substack{i \in N_1 \\ f_i > f_0}} (\lfloor \alpha_i \rfloor + 1) x_i \leq \beta + \sum_{i \in N_2} \alpha_i^- y_i + \sum_{\substack{i \in N_1 \\ f_i > f_0}} (1 - f_i) x_i.$$

Se ora indichiamo con z il primo membro e con s il secondo membro (eccetto β), possiamo scrivere l'ultima disuguaglianza in modo più conciso: $z \leq \beta + s$.

Poiché z deve assumere valore intero e $s \geq 0$, possiamo applicare il Lemma 2.8 per dedurre che la disuguaglianza $z \leq \beta + \frac{s}{1-f_0}$ è valida per la nostra regione ammissibile. Riscritta nelle variabili originali e dopo qualche piccola manipolazione algebrica, tale disequazione coincide con la (2.46).

Resta da mostrare che $(\bar{x}, \bar{y})$ non soddisfa (2.46). Questo segue immediatamente dal fatto che $\bar{x}_i = 0$ per ogni $i \in N_1$, $\bar{y}_i = 0$ per ogni $i \in N_2$ e $\bar{x}_j = \beta > \lfloor \beta \rfloor$. □

La disequazione (2.46) è nota come *taglio di Gomory intero misto*. Si osservi che, con una piccola modifica (in realtà semplificazione) della dimostrazione precedente, si può vedere che la disuguaglianza (2.46) è un piano di taglio anche se come coefficiente di ogni x_i scriviamo $\lfloor \alpha_i \rfloor$. Tuttavia, la disequazione nella forma data sopra è più forte.

Purtroppo nel caso intero misto non vale un risultato di convergenza analogo a quello citato sopra per il caso puro: un algoritmo che ad ogni iterazione aggiunge un taglio di Gomory intero misto non converge, in generale, alla soluzione ottima del programma lineare intero misto. Ciononostante, questi tagli si rivelano molto utili nella pratica.

2.4.5 Branch-and-cut

Il branch-and-bound e l'approccio per piani di taglio presentano un problema comune: accade spesso che le prime iterazioni portino ad una rapida riduzione della distanza dalla soluzione ottima intera, ma che in seguito ogni iterazione porti ad un miglioramento irrisorio (o addirittura nullo) della soluzione. Un utilizzo combinato dei due algoritmi, detto *branch-and-cut*, può aiutare ad attenuare questo problema: sostanzialmente, durante la costruzione del'albero di branch-and-bound, si migliora il rilassamento lineare del sottoproblema corrente aggiungendo qualche piano di taglio.

2.5 Matrici totalmente unimodulari

Come già osservato in più occasioni, risolvendo il rilassamento lineare di un programma lineare intero non si trova, in generale, una soluzione ottima intera. Studiamo ora alcune condizioni che invece garantiscono che ciò accada. Si tenga presente che quanto verrà detto in questa sezione riguarda esclusivamente il caso della programmazione lineare intera *pura*.

2.5.1 Forma standard: matrici unimodulari

Una matrice $A \in \mathbb{Z}^{m \times n}$, con $m \leq n$, è detta *unimodulare* se ogni sottomatrice $m \times m$ di A ha determinante uguale a 0, 1 o -1.

Proposizione 2.10 *Data una matrice $A \in \mathbb{Z}^{m \times n}$, le soluzioni ammissibili di base del sistema in forma standard*

$$Ax = b$$
$$x \geq \mathbf{0}$$

sono tutte intere qualunque sia $b \in \mathbb{Z}^m$ se e solo se A è unimodulare.

Dimostrazione. Assumiamo che A sia unimodulare e consideriamo una soluzione di base $\bar{x}$ associata ad una qualche base $B \subseteq \{1, \ldots, n\}$, cioè $\bar{x}_B = A_B^{-1} b$, $\bar{x}_N = \mathbf{0}$. Poiché A è unimodulare e A_B è invertibile, $|\det A_B| = 1$. La regola di inversione delle matrici mostra immediatamente che allora A_B^{-1} ha tutte le componenti intere, da cui segue che $\bar{x}$ è un vettore intero qualunque sia $b \in \mathbb{Z}^m$.

Per mostrare l'implicazione inversa, fissiamo una sottomatrice M di A di dimensione $m \times m$ e mostriamo che se $\det M \neq 0$ allora $\det M \in \{-1, 1\}$. Se $\det M \neq 0$, allora $M = A_B$ per una qualche base B. Fissiamo un qualunque vettore $b \in \mathbb{Z}^m$ e sia $\bar{x}$ la corrispondente soluzione di base.[5] Scegliamo un qualsiasi vettore intero $\bar{y}_B$ tale che $\bar{x}_B + \bar{y}_B \geq \mathbf{0}$ e definiamo $b' = A_B(\bar{x}_B + \bar{y}_B)$. Poiché b' è un vettore intero, per ipotesi tutte le soluzioni ammissibili di base del sistema

$$\begin{aligned} Az &= b' \\ z &\geq \mathbf{0} \end{aligned}$$

sono intere. Siccome la soluzione associata alla base B è $\bar{z}_B = A_B^{-1} b' = \bar{x}_B + \bar{y}_B \geq \mathbf{0}$ ed è dunque ammissibile, possiamo allora asserire che $\bar{z}_B$ è un vettore intero. Poiché $\bar{z}_B = \bar{x}_B + \bar{y}_B$ e $\bar{y}_B$ è anch'esso intero, deduciamo che $\bar{x}_B$ è un vettore intero. Siccome $\bar{x}_B = A_B^{-1} b$ ed il vettore b era arbitrario, abbiamo mostrato che $A_B^{-1} b$ è un vettore intero qualunque sia $b \in \mathbb{Z}^m$. Scegliendo in particolare $b = e_j$ (per ogni possibile j), scopriamo che ogni colonna di A_B^{-1} è un vettore intero, cioè tutte le componenti di A_B^{-1} sono intere. Allora, dato che $(\det A_B)(\det A_B^{-1}) = 1$ ed entrambi i fattori sono numeri interi, si ha necessariamente $|\det A_B| = 1$. □

Come conseguenza di questo risultato, dato un programma lineare intero in forma standard

$$\begin{aligned} \max\ & c^\top x \\ \text{s.a}\ & Ax = b \\ & x \geq \mathbf{0} \\ & x \in \mathbb{Z}^n \end{aligned}$$

con A unimodulare e b intero, applicando il metodo del simplesso al rilassamento lineare del problema si trova una soluzione ottima intera.

2.5.2 Forma generica: matrici totalmente unimodulari

Una matrice si dice *totalmente unimodulare* se tutte le sue sottomatrici quadrate (di ogni ordine) hanno determinante uguale a 0, 1 o -1. Si noti che in particolare tutte le componenti di una matrice totalmente unimodulare sono uguali a 0, 1 o -1. Dalla definizione segue anche che ogni sottomatrice di una matrice totalmente unimodulare è ancora totalmente unimodulare.

È facile verificare le seguenti equivalenze:

[5] Si noti che non sappiamo se $\bar{x}$ è ammissibile, dunque non possiamo usare l'ipotesi per dire che $\bar{x}$ è un vettore intero.

Osservazione 2.11 *Le seguenti proprietà sono equivalenti:*

(a) *A è totalmente unimodulare;*

(b) *$A^\top$ è totalmente unimodulare;*

(c) *$[A \mid e_j]$ è totalmente unimodulare per j qualunque;*

(d) *$[A \mid \alpha]$ è totalmente unimodulare, dove α è una qualunque colonna di A;*

(e) *la matrice ottenuta cambiando di segno una riga o colonna di A è totalmente unimodulare.*

Lemma 2.12 *Una matrice A di dimensioni $m \times n$ è totalmente unimodulare se e solo $[A \mid I_m]$ è unimodulare, dove I_m è la matrice identità $m \times m$.*

Dimostrazione. Se A è totalmente unimodulare, applicando ripetutamente la proprietà (c) dell'Osservzione 2.11 si dimostra che $[A \mid I_m]$ è totalmente unimodulare, dunque anche unimodulare.

Supponiamo ora che $[A \mid I_m]$ sia unimodulare e prendiamo una qualunque sottomatrice quadrata M di A. Dobbiamo mostrare che $\det M \in \{0, \pm 1\}$. Siano R e C gli insiemi degli indici delle righe e colonne (rispettivamente) di A che compongono M. Chiaramente $|R| = |C| \leq m$. Estendiamo M ad una sottomatrice $m \times m$ di $[A \mid I_m]$ selezionando, oltre alle colonne di A con indice in C, le colonne di I_m che hanno il loro 1 su una riga non in R. Detta M' la matrice così ottenuta, l'unimodularità di A dà $\det M' \in \{0, \pm 1\}$. D'altra parte, sviluppando il determinante di M' con il metodo di Laplace, si ottiene $|\det M'| = |\det M|$, da cui segue che $\det M \in \{0, \pm 1\}$. □

Nel teorema e nel corollario seguenti, quando ci riferiamo alle "soluzioni di base" di un sistema che non è in forma standard, intendiamo le soluzioni di base del sistema che si ottiene passando alla forma standard.

Teorema 2.13 (Hoffman e Kruskal 1956) *Data una matrice $A \in \mathbb{Z}^{m \times n}$, le soluzioni ammissibili di base del sistema*

$$Ax \leq b \tag{2.47}$$
$$x \geq \mathbf{0} \tag{2.48}$$

sono tutte intere qualunque sia $b \in \mathbb{Z}^m$ se e solo se A è totalmente unimodulare.

Dimostrazione. Messo in forma standard, il sistema (2.47)–(2.48) si scrive

$$Ax + s = b$$
$$x \geq \mathbf{0},\ s \geq \mathbf{0},$$

dove s è il vettore delle variabili scarto. Per la Propisizione 2.10, le soluzioni ammissibili di base di questo sistema sono tutte intere qualunque sia $b \in \mathbb{Z}^m$ se e solo se la matrice dei vincoli, cioè $[A \mid I_m]$, è unimodulare. Per il lemma precedente, questo avviene se e solo se A è totalmente unimodulare. □

Corollario 2.14 *Data una matrice $A \in \mathbb{Z}^{m\times n}$, le soluzioni ammissibili di base del sistema $Ax \sim b$ sono tutte intere qualunque sia $b \in \mathbb{Z}^m$ se e solo se A è totalmente unimodulare.*

Dimostrazione. Scriviamo il sistema $Ax \sim b$ nella forma (2.47)–(2.48), così da poter applicare il Teorema 2.13. Come noto dalla programmazione lineare, per farlo è necessario

- sostituire le eventuali equazioni con coppie di disequazioni opposte,
- cambiare il segno alle eventuali disequazioni del tipo $a^\top x \geq \beta$,
- sostituire ogni variabile x_j non esplicitamente vincolata ad essere non-negativa con l'espressione $x'_j - x''_j$, dove x'_j, x''_j sono due nuove variabili, per le quali va imposta la non-negatività.

Sia A' la matrice del sistema così ottenuto. Per il Teorema 2.13, le soluzioni ammissibili di base sono tutte intere qualunque sia b intero se e solo se A' è totalmente unimodulare. Ma utilizzando ripetutamente le equivalenze elencate nell'Osservazione 2.11, si verifica che A' è totalmente unimodulare se e solo se anche A lo è. □

Come conseguenza di questo risultato, dato un programma lineare intero nella generica forma

$$\begin{aligned} \max\ & c^\top x \\ \text{s.a}\ & Ax \sim b \\ & x \in \mathbb{Z}^n \end{aligned}$$

con A totalmente unimodulare e b intero, applicando il metodo del simplesso al rilassamento lineare del problema si trova una soluzione ottima intera. Si tenga comunque presente che questa eventualità potrebbe verificarsi anche nel caso in cui A non sia totalmente unimodulare.

2.5.3 Riconoscere le matrici totalmente unimodulari

Data una matrice $A \in \{0, \pm 1\}^{m\times n}$, una *bicolorazione equa* delle colonne di A è una partizione dell'insieme $\{1, \ldots, n\}$ in due sottoinsiemi R, B tali che

$$\sum_{j\in R} a_{ij} - \sum_{j\in B} a_{ij} \in \{0, \pm 1\}, \quad i = 1, \ldots, m. \tag{2.49}$$

Spesso ci si riferisce alle colonne con indici in R o B come colonne *rosse* e *blu* rispettivamente. La condizione (2.49) può essere così riformulata: per ogni riga di A, la somma degli elementi rossi deve uguagliare quella degli elementi blu nel caso in cui la somma di tutti gli elementi della riga sia pari, mentre in caso contrario la somma degli elementi rossi e la somma degli elementi blu devono differire di esattamente una unità.

Una bicolorazione delle colonne di A può essere vista come un vettore $z \in \{-1, 1\}^n$ che indica se la j-esima colonna riceve colore rosso ($z_j = 1$) o blu

($z_j = -1$). La condizione (2.49) può allora essere scritta nella forma compatta $-\mathbf{1} \le Az \le \mathbf{1}$.

In alternativa, definendo $x_j = 1$ se la colonna j viene colorata di rosso e $x_j = 0$ altrimenti, l'insieme di tutte le possibili bicolorazioni eque delle colonne di A è descritto dai vincoli

$$\left\lfloor \frac{1}{2} A\mathbf{1} \right\rfloor \le Ax \le \left\lceil \frac{1}{2} A\mathbf{1} \right\rceil \tag{2.50}$$

$$x \in \{0,1\}^n, \tag{2.51}$$

dove il simbolo di arrotondamento per difetto o per eccesso applicato ad un vettore indica che l'operazione viene svolta su tutte le componenti del vettore. Per convincersi della correttezza dei vincoli, basta osservare che ogni componente del vettore Ax è uguale alla somma degli elementi rossi della corrispondente riga di A, mentre ogni componente del vettore $A\mathbf{1}$ è la somma di tutte le componenti della corrispondente riga di A. I vincoli dicono allora che la somma degli elementi rossi di ogni riga deve differire di meno di un'unità dalla semisomma di tutti gli elementi della riga stessa. Ma questo equivale alla definizione di bicolorazione equa.

Teorema 2.15 (Ghouila-Houri 1962) *Una matrice $A \in \{0, \pm 1\}^{m \times n}$ è totalmente unimodulare se e solo se ogni sottomatrice di A ammette una bicolorazione equa delle sue colonne.*

Dimostrazione. Supponiamo che A sia totalmente unimodulare. Poiché tutte le sottomatrici di A sono totalmente unimodulari, sarà sufficiente mostrare che esiste una bicolorazione equa delle colonne di A. Come osservato sopra, questo equivale a dire che la regione definita da (2.50)–(2.51) ha almeno una soluzione ammissibile. Il rilassamento lineare di (2.50)–(2.51), ottenuto scrivendo $\mathbf{0} \le x \le \mathbf{1}$ invece di $x \in \{0,1\}^n$, è descritto da una matrice totalmente unimodulare e da un termine noto intero. Allora il Corollario 2.14 implica che tutte le soluzioni ammissibili di base di tale rilassamento lineare sono intere. Per concludere che esiste una bicolorazione equa delle colonne di A (cioè una soluzione di (2.50)–(2.51)), dobbiamo assicurarci che il rilassamento lineare non sia vuoto: solo in tal caso, infatti, l'esistenza di una soluzione ammissibile di base è garantita. Ma una soluzione ammissibile per il rilassamento lineare è data da $x_j = 1/2$ per ogni $j = 1, \ldots, n$.

Per l'implicazione inversa, prendiamo una sottomatrice $k \times k$ di A, chiamiamola M, e mostriamo che se $\det M \neq 0$ allora $\det M \in \{-1, 1\}$. La dimostrazione è per induzione su k. Il risultato è chiaramente vero per $k = 1$. Sia dunque $k \ge 2$ ed assumiamo che la proprietà sia vera per tutte le sottomatrici di ordine $(k-1) \times (k-1)$. Definiamo $\Delta = \det M \neq 0$ (si noti che Δ è un intero). La regola di inversione delle matrici e l'ipotesi induttiva implicano che tutte le componenti di M^{-1} appartengono a $\{0, \pm 1/\Delta\}$. Allora, detta c la prima colonna della matrice ΔM^{-1}, abbiamo $c \in \{0, \pm 1\}^n$ e $Mc = \Delta e_1$. Detto S il supporto di c, cioè $S = \{j : c_j \neq 0\}$, si ha $S \neq \varnothing$ e $M_S c_S = \Delta e_1$. Come osservato più sopra, l'esistenza di una bicolorazione equa delle colonne di M_S (garantita dall'ipotesi) significa che esiste un vettore $z_S \in \{-1, 1\}^S$ tale che $-\mathbf{1} \le M_S z_S \le \mathbf{1}$. Ora distinguiamo due casi.

1. Se Δ è pari, le condizioni $M_S c_S = \Delta e_1$ e $c_S \in \{-1, 1\}^S$ implicano che la somma degli elementi di ciascuna riga di M_S è pari. Allora in tal caso si deve avere $M_S z_S = \mathbf{0}$. Completando z_S con componenti nulle, otteniamo un vettore $z \in \{0, \pm 1\}^k$ tale che $Mz = \mathbf{0}$. Siccome $\det M \neq 0$, questo implica che $z = \mathbf{0}$, contraddicendo il fatto che $z_j \in \{-1, 1\}$ per $j \in S$.

2. Se Δ è dispari, ragionando come sopra si conclude che esiste $z \in \{0, \pm 1\}^k$ tale che $Mz = e_1$. Poiché anche $M\frac{c}{\Delta} = e_1$ e $\det M \neq 0$, abbiamo $z = \frac{c}{\Delta}$. Siccome entrambi i vettori z e c hanno tutte le componenti uguali a 0, 1 o -1 (ed almeno una componente diversa da 0), si deve necessariamente avere $|\Delta| = 1$.

□

Si noti che nel Teorema 2.15 ci si può restringere alle sole sottomatrici di A con m righe (cioè quelle in cui tutte le righe di A vengono selezionate). Inoltre, dato che una matrice è totalmente unimodulare se e solo se tale è la sua trasposta, si può definire il concetto simmetrico di bicolorazione delle righe ed enunciare un analogo teorema in cui i ruoli di righe e colonne sono invertiti.

Matrici con al massimo due elementi non-nulli per riga o per colonna

Sia $A \in \{0, \pm 1\}^{m \times n}$ una matrice con al massimo due elementi non-nulli per riga. In questo caso la condizione (2.49), che definisce una bicolorazione equa delle colonne di A, può essere così riformulata: per ogni riga di A con esattamente due elementi non-nulli, tali elementi hanno lo stesso colore se sono di segno opposto e colore diverso se sono dello stesso segno. Inoltre, è immediato verificare che una bicolorazione equa delle colonne di una tale matrice induce una bicolorazione equa delle colonne di una sua qualunque sottomatrice. Otteniamo allora il seguente corollario del Teorema 2.15:

Corollario 2.16 *Una matrice $A \in \{0, \pm 1\}^{m \times n}$ con al massimo due elementi non-nulli per riga (rispettivamente colonna) è totalmente unimodulare se e solo se ammette una bicolorazione equa delle sue colonne (rispettivamente righe).*

2.5.4 Esempi

Problema di assegnamento e problema dei trasporti con merce indivisibile

Mostriamo che la formulazione per il problema di assegnamento vista nella Sezione 2.2.2 è descritta da una matrice totalmente unimodulare. Il rilassamento lineare della regione ammissibile è il seguente:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n \tag{2.52}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n \tag{2.53}$$

$$0 \leq x_{ij} \leq 1, \quad i, j = 1, \ldots, n. \tag{2.54}$$

Grazie all'Osservazione 2.11, nel decidere se la matrice di questo sistema è totalmente unimodulare possiamo ignorare le righe corrispondenti ai vincoli (2.54), in quanto ciascuna di esse contiene un unico elemento diverso da zero. Nella matrice residua, ogni variabile x_{ij} ha coefficiente 1 in esattamente un vincolo (2.52) ed un vincolo (2.53), mentre tutti gli altri coefficienti sono nulli. Dunque ogni colonna contiene esattamente due elementi non-nulli, entrambi uguali a 1, uno nel primo gruppo di righe e l'altro nel secondo. Se coloriamo di rosso le righe del primo gruppo e di blu quelle del secondo, la condizione del Corollario 2.16 è soddisfatta, dunque la matrice è totalmente unimodulare.

Un ragionamento del tutto analogo mostra che la matrice del modello scritto nella Sezione 1.4.3 per il problema dei trasporti è totalmente unimodulare. Dunque se richiediamo l'interezza delle variabili (ad esempio perché la merce da trasportare è indivisibile) ed abbiamo termini noti interi, il metodo del simplesso ci darà direttamente una soluzione ottima intera.

Problema del cammino minimo

Nota Nel seguito verrà usata la terminologia di base della teoria dei grafi. Inoltre verranno discussi alcuni problemi fondamentali di ottimizzazione su grafi, ma la trattazione sarà limitata alle connessioni con la programmazione lineare intera. Maggiori dettagli possono essere trovati, ad esempio, nelle dispense del corso di Ottimizzazione Discreta.

Sia $G = (V, E)$ un grafo orientato, dove V è l'insieme dei nodi ed E è l'insieme degli archi. Supponiamo che per ogni arco $e = (i, j) \in E$ sia dato un costo c_e. Dato un sottoinsieme di archi $F \subseteq E$, il costo di F è definito come $c(F) = \sum_{e \in F} c_e$. Per motivi che diventeranno chiari più avanti, assumiamo che i costi siano *conservativi*, cioè che non esista alcun ciclo orientato di costo negativo in G. Dati due nodi $s, t \in V$, il problema del *cammino minimo* consiste nel trovare un cammino orientato da s a t che abbia costo minimo.

Per ogni arco $e \in E$, definiamo una variabile binaria x_e che vale 1 se e solo se l'arco e fa parte del cammino minimo. Si noti che, per ogni nodo $i \in V \setminus \{s, t\}$, vi sono due possibilità: o nessuno degli archi incidenti in i fa parte del cammino minimo, o il cammino minimo contiene esattamente un arco entrante in i ed uno uscente da i. In entrambi i casi, il numero di archi del cammino minimo entranti in i è uguale al numero di archi del cammino minimo uscenti da i, condizione che può essere scritta così:

$$\sum_{e \in \delta^-(i)} x_e - \sum_{e \in \delta^+(i)} x_e = 0,$$

dove $\delta^-(i)$ e $\delta^+(i)$ indicano rispettivamente l'insieme di archi di G entranti ed uscenti da i. Per i nodi s e t, invece, il secondo membro dovrà essere uguale rispettivamente a -1 e 1. Definiamo allora

$$b_i = \begin{cases} 0 & \text{per } i \in V \setminus \{s, t\} \\ -1 & \text{per } i = s \\ +1 & \text{per } i = t \end{cases}$$

e scriviamo la seguente formulazione:

$$\min \sum_{e \in E} c_e x_e \tag{2.55}$$

$$\text{s.a} \sum_{e \in \delta^-(i)} x_e - \sum_{e \in \delta^+(i)} x_e = b_i, \quad i \in V \tag{2.56}$$

$$x_e \in \{0,1\}, \qquad e \in E. \tag{2.57}$$

Si noti che tale formulazione accetta come soluzioni ammissibili anche sequenze di archi che passano più di una volta per lo stesso nodo (mentre per un cammino questo non è consentito). Tuttavia, quando questo avviene la soluzione in questione è un insieme di archi che contiene un ciclo C. Poiché abbiamo inizialmente assunto che non ci siano cicli di costo negativo, è immediato verificare che rimuovendo gli archi di C dalla soluzione si ottiene una soluzione ammissibile migliore. Iterando questo procedimento, si arriva a dimostrare che la soluzione ottima di (2.55)–(2.57) è effettivamente un cammino da s a t.

Definiamo la matrice di incidenza di G come una matrice A in cui ogni riga corrisponde ad un nodo di G ed ogni colonna ad un arco di G, con

$$a_{i,e} = \begin{cases} -1 & \text{se l'arco } e \text{ esce dal nodo } i \\ +1 & \text{se l'arco } e \text{ entra nel nodo } i \\ 0 & \text{altrimenti.} \end{cases}$$

Il problema (2.55)–(2.57) può allora essere scritto come segue:

$$\begin{aligned} \min\ & c^\top x \\ \text{s.a}\ & Ax = b \\ & x \in \{0,1\}^E. \end{aligned}$$

Mostriamo ora che A è una matrice totalmente unimodulare. Poiché ogni colonna di A contiene esattamente due elementi diversi da zero (un $+1$ ed un -1), possiamo usare il Corollario 2.16, colorando tutte le righe dello stesso colore. Dunque A è totalmente unimodulare, da cui segue che un cammino minimo può essere trovato risolvendo il rilassamento lineare

$$\begin{aligned} \min\ & c^\top x \\ \text{s.a}\ & Ax = b \\ & 0 \le x \le \mathbf{1}. \end{aligned}$$

Problema del flusso massimo

Sia dato un grafo orientato $G = (V, E)$ e supponiamo che ogni arco $e \in E$ abbia una *capacità* $u_e > 0$. Siano inoltre s e t due nodi fissati. Un *flusso* in G è un vettore $x \in \mathbb{R}^E$ tale che:

- $0 \le x_e \le u_e$ per ogni $e \in E$;

- *conservazione del flusso*: per ogni $i \in V \setminus \{s,t\}$, il flusso entrante in i è uguale a quello uscente da i, cioè

$$\sum_{e \in \delta^-(i)} x_e - \sum_{e \in \delta^+(i)} x_e = 0.$$

Il problema del *flusso massimo* consiste nel trovare un flusso di valore massimo in G, dove il valore di un flusso è definito come la quantità totale uscente da s (o, equivalentemente, entrante in t).

Per scrivere una formulazione per questo problema, conviene aggiungere al grafo l'arco (t,s), a cui associamo una variabile ϕ. Possiamo allora imporre la conservazione del flusso anche per i nodi s e t. Inoltre, il valore del flusso sarà esattamente quello che scorre sull'arco (t,s). Otteniamo dunque

$$\max \quad \phi \tag{2.58}$$

$$\text{s.a} \quad \sum_{e \in \delta^-(i)} x_e - \sum_{e \in \delta^+(i)} x_e = 0, \qquad i \in V \setminus \{s,t\} \tag{2.59}$$

$$\sum_{e \in \delta^-(s)} x_e - \sum_{e \in \delta^+(s)} x_e + \phi = 0 \tag{2.60}$$

$$\sum_{e \in \delta^-(t)} x_e - \sum_{e \in \delta^+(t)} x_e - \phi = 0 \tag{2.61}$$

$$x_e \le u_e, \qquad e \in E \tag{2.62}$$

$$x_e \ge 0, \qquad e \in E. \tag{2.63}$$

Analogamente a quanto visto per il problema del cammino minimo, la matrice di questo sistema è totalmente unimodulare. Ne segue che se le capacità sono tutte intere, esiste una soluzione ottima intera, che verrà trovata dal metodo del simplesso.

Problema del taglio minimo

Come sopra, abbiamo un grafo orientato $G = (V,E)$ con capacità $u_e > 0$ su ogni arco $e \in E$ e sono fissati due nodi s e t. Dato un sottoinsieme $S \subseteq V$ tale che $s \in S$ e $t \notin S$, si definisce il *taglio* $\delta(S) = \{(i,j) \in E : i \in S, j \notin S\}$. La capacità del taglio è definita come la somma delle capacità degli archi che lo compongono. Il problema del *taglio minimo* chiede di trovare un taglio in G che abbia capacità minima.

Vogliamo mostrare che questo problema è essenzialmente il duale del problema del flusso massimo. Associamo dunque variabili duali v_i, v_s e v_t rispettivamente ai vincoli (2.59), (2.60) e (2.61), e z_e ai vincoli (2.62). Il duale del programma lineare (2.58)–(2.63) è il seguente:

$$\min \quad \sum_{e \in E} u_e z_e \tag{2.64}$$

$$\text{s.a} \quad -v_i + v_j + z_e \ge 0, \quad e = (i,j) \in E \tag{2.65}$$

$$-v_t + v_s \ge 1, \tag{2.66}$$

$$z_e \ge 0, \qquad e \in E. \tag{2.67}$$

Poiché la matrice del sistema è totalmente unimodulare (in quanto è la trasposta della matrice del problema (2.58)–(2.63)), esiste una soluzione ottima intera. In realtà è possibile dimostrare che esiste una soluzione ottima con tutte le componenti in $\{0, 1\}$. Possiamo allora interpretare questo modello come una formulazione per il problema del taglio minimo. Il significato delle variabili è il seguente: per ogni $i \in V$, $v_i = 1$ se e solo se $i \in S$, mentre per ogni $e \in E$, $z_e = 1$ se e solo se $e \in \delta(S)$. Si noti che la condizione (2.66) assicura che $v_s = 1$ e $v_t = 0$, cioè $s \in S$ e $t \notin S$, come richiesto dalla definizione di taglio. Inoltre, poiché il problema è di minimizzazione e i coefficienti delle variabili z_e nella funzione obiettivo sono tutti positivi, i vincoli (2.65) fanno sì che in una soluzione ottima $z_e = 1$ esattamente quando $v_i = 1$ e $v_j = 0$, cioè quando $i \in S$ e $j \notin S$. Dunque il modello ottenuto formula correttamente il problema del taglio minimo.

Osserviamo anche che, grazie al Teorema di Dualità Forte (Teorema 1.5), la relazione di dualità appena osservata implica che il valore di un flusso massimo in G è uguale alla minima capacità di un taglio in G (risultato ottenuto da Ford e Fulkerson nel 1956 senza far uso della programmazione lineare).

Problema del flusso di costo minimo

Sia dato un grafo orientato $G = (V, E)$ con capacità $u_e > 0$ e costi c_e sugli archi. Supponiamo che ad ogni nodo i sia associato un bilancio b_i. Ridefiniamo un *flusso* come un vettore $x \in \mathbb{R}^E$ tale che

- $0 \leq x_e \leq u_e$ per ogni $e \in E$;
- $\sum_{e \in \delta^-(i)} x_e - \sum_{e \in \delta^+(i)} x_e = b_i$ per ogni $i \in V$.

Il problema del *flusso di costo minimo* chiede di determinare un flusso x che minimizzi il costo totale, definito come $\sum_{e \in E} c_e x_e$.

Il problema può essere formulato come programma lineare:

$$\min\ c^\top x \tag{2.68}$$
$$\text{s.a}\ Ax = b \tag{2.69}$$
$$0 \leq x \leq u, \tag{2.70}$$

dove A è la matrice di incidenza di G. La totale unimodularità di A implica che, se capacità e bilanci sono tutti interi, troveremo un flusso ottimo intero (sempre che esista almeno un flusso ammissibile).

Si noti che le formulazioni date sopra per il problema del cammino minimo e per quello del flusso massimo sono casi particolari di (2.68)–(2.70).

Problema del matching massimo su grafo bipartito

Sia $G = (V, E)$ un grafo non orientato. Un *matching* in G è un sottoinsieme di archi $M \subseteq E$ tale che ogni nodo di G sia un'estremità di al massimo un arco in M. In altre parole, non esistono due archi in M con un'estremità in comune.

Associando una variabile binaria x_e ad ogni arco $e \in E$, il problema di trovare il matching di cardinalità massima può essere formulato così:

$$\begin{aligned} \max \ & \sum_{e \in E} x_e \\ \text{s.a} \ & \sum_{e \in \delta(i)} x_e \leq 1, \quad i \in V \\ & x_e \in \{0,1\}, \qquad e \in E, \end{aligned}$$

dove $\delta(i)$ indica l'insieme di archi incidenti in nel nodo i. Possiamo definire la *matrice di incidenza* di G in modo analogo a quanto fatto per i grafi orientati: per un grafo non orientato G, la matrice di incidenza A ha una riga per ogni nodo di G ed una colonna per ogni arco di G, con

$$a_{i,e} = \begin{cases} 1 & \text{se l'arco } e \text{ è incidente in } i \\ 0 & \text{altrimenti.} \end{cases}$$

La formulazione per il problema del matching massimo può allora essere scritta in modo più compatto:

$$\begin{aligned} \max \ & \mathbf{1}^\top x \\ \text{s.a} \ & Ax \leq \mathbf{1} \\ & x \in \{0,1\}^E. \end{aligned}$$

La matrice A non è, in generale, totalmente unimodulare (si veda il caso di un grafo completo con tre nodi). Tuttavia, come mostriamo ora, A è totalmente unimodulare quando G è un grafo *bipartito*. Ricordiamo che G si dice bipartito se l'insieme dei nodi V può essere partizionato in due sottoinsiemi V_1, V_2 in modo tale che ogni arco del grafo abbia un'estremità in V_1 e l'altra in V_2. In tal caso, possiamo applicare il Corollario 2.16, colorando di rosso le righe di A corrispondenti a nodi in V_1 e di blu quelle corrispondenti a nodi in V_2. Questo implica che se G è bipartito allora A è totalmente unimodulare, dunque un matching massimo può essere trovato risolvendo il programma lineare

$$\max \ \mathbf{1}^\top x \tag{2.71}$$

$$\text{s.a} \ Ax \leq \mathbf{1} \tag{2.72}$$

$$x \geq \mathbf{0} \tag{2.73}$$

(la condizione $x \leq \mathbf{1}$ è implicata dagli altri vincoli).

Osserviamo che il problema del matching massimo su grafo bipartito può essere formulato anche come problema di flusso massimo. A questo scopo, è necessario introdurre due nodi aggiuntivi s e t, creare archi (s,i) per ogni $i \in V_1$ e (j,t) per ogni $j \in V_2$, ed orientare tutti gli archi originali di G da V_1 a V_2. Inoltre attribuiamo capacità 1 a tutti gli archi del grafo orientato così ottenuto. Poiché le capacità sono intere, esiste un flusso massimo intero. Questo significa che su ogni arco scorre un flusso pari a 0 o 1. Si verifica facilmente che gli archi originali di G utilizzati da un flusso massimo intero formano un matching massimo in G e che il numero di archi del matching è uguale al valore del flusso.

Notiamo infine che il problema del matching massimo (su grafo generico) è un caso particolare del problema di set packing (Sezione 2.2.6).

Problema del vertex cover minimo su grafo bipartito

Il duale del programma lineare (2.71)–(2.73) è il seguente:

$$\begin{aligned} \min\ & \mathbf{1}^\top u \\ \text{s.a}\ & A^\top u \geq \mathbf{1} \\ & u \geq \mathbf{0}. \end{aligned}$$

Poiché, se G è bipartito, $A^\top$ è totalmente unimodulare, questo problema ha una soluzione ottima intera. È facile verificare che in realtà questa soluzione ottima deve avere tutte le componenti in $\{0, 1\}$ (se così non fosse, otterremmo una soluzione ammissibile migliore diminuendo a 1 il valore delle componenti di u maggiori di 1). Una soluzione ottima $\bar{u} \in \{0, 1\}^V$ può essere interpretata come un *vertex cover* di cardinalità minima. Per vertex cover si intende un sottoinsieme di vertici $C \subseteq V$ tale che ogni arco del grafo abbia almeno un'estremità in C. Grazie al Teorema di Dualità Forte (Teorema 1.5), scopriamo dunque che per grafi bipartiti la massima cardinalità di un matching è uguale alla minima cardinalità di un vertex cover (risultato noto come Teorema di König e dimostrato per altra via nel 1931, ben prima della nascita della programmazione lineare).

Si noti che il problema del vertex cover minimo (su grafo generico) è un caso particolare del problema di set covering (Sezione 2.2.6).

Problema del matching perfetto di costo minimo su grafo bipartito

Consideriamo infine il problema del *matching perfetto* di costo minimo (o massimo). Un matching si dice perfetto se ogni nodo di G è estremità di esattamente un arco del matching. Se sono assegnati costi c_e agli archi, il problema chiede di determinare un matching perfetto (se ne esiste uno) tale che il costo totale degli archi scelti sia il minimo (o massimo) possibile. Se A è la matrice di incidenza di G, una formulazione per questo problema è la seguente:

$$\begin{aligned} \min\ (\text{o max})\ & \mathbf{1}^\top x \\ \text{s.a}\ & Ax = \mathbf{1} \\ & x \in \{0, 1\}^E. \end{aligned}$$

Se G è bipartito, allora A è totalmente unimodulare e i vincoli di interezza possono essere rilassati.

Il problema del matching perfetto di costo minimo può essere formulato come problema di flusso di costo minimo: basta orientare tutti gli archi di G da V_1 a V_2, assegnare capacità 1 a tutti gli archi e definire bilanci $b_i = -1$ per $i \in V_1$ e $b_i = 1$ per $i \in V_2$. L'interezza di capacità e bilanci implica che se c'è una soluzione ottima, allora ce n'è una in cui il flusso su ogni arco è 0 o 1. Gli archi su cui scorre flusso di valore 1 individuano un matching perfetto di costo minimo. Si noti anche che questo problema non è altro che il problema di assegnamento.

Concludiamo osservando che il problema del matching perfetto di costo minimo o massimo (su grafo generico) è un caso particolare del problema di set partitioning (Sezione 2.2.6).

2.6 Problemi con molti vincoli: generazione di righe

Abbiamo visto nelle sezioni precedenti che risolvere il rilassamento lineare costituisce il primo passo di vari metodi risolutivi per la programmazione lineare intera (mista). Tuttavia in alcuni casi il rilassamento lineare di un problema è già esso stesso troppo grosso per essere risolto direttamente con il metodo del simplesso o con un altro algoritmo. In tali situazioni, un approccio intelligente consiste, quando possibile, nel partire con un problema ridotto (non comprendente, cioè, tutti i vincoli e le variabili del problema originale) ed aggiungere vincoli o variabili "all'occorrenza". In questa sezione ci occupiamo del caso in cui il programma lineare intero dato (e dunque il suo rilassamento lineare) contenga un numero enorme di vincoli. L'idea di base sarà illustrata per mezzo del *problema del commesso viaggiatore*.

2.6.1 Problema del commesso viaggiatore

Sia $G = (V, E)$ un grafo orientato completo (contenente, cioè, archi (i, j) e (j, i) per ogni coppia di nodi distinti $i, j \in V$) e supponiamo che sia assegnato un costo c_e per ogni arco $e \in E$. Il problema del commesso viaggiatore[6] consiste nel trovare un *ciclo hamiltoniano* di costo minimo in G, dove un ciclo si dice hamiltoniano se passa una ed una sola volta per ciascun nodo. L'assunzione che il grafo sia completo può essere fatta senza perdita di generalità: se così non fosse, basterebbe aggiungere gli archi mancanti ed assegnare loro un costo abbastanza alto.

Formulazione

Vogliamo formulare questo problema come programma lineare intero. Per ogni arco $e \in E$, introduciamo una variabile binaria x_e che vale 1 se e solo se l'arco e fa parte del ciclo scelto. Poiché ogni ciclo hamiltoniano entra ed esce esattamente una volta da ciascun nodo, possiamo imporre i vincoli

$$\sum_{e \in \delta^-(i)} x_e = 1, \quad i \in V \tag{2.74}$$

$$\sum_{e \in \delta^+(i)} x_e = 1, \quad i \in V. \tag{2.75}$$

Questi vincoli, però, accettano come soluzioni ammissibili anche insiemi di archi che chiudono cicli parziali: ad esempio, se $V = \{1, 2, 3, 4, 5, 6, 7\}$, la soluzione $x_{12} = x_{23} = x_{31} = x_{45} = x_{56} = x_{67} = x_{74} = 1$ (e tutte le altre componenti uguali a zero) risulta ammissibile, anche se non corrisponde ad un ciclo hamiltoniano. In generale, è facile convincersi che un vettore $x \in \{0, 1\}^E$ soddisfa (2.74)–(2.75) se e solo se il corrispondente insieme di archi è un'unione di cicli che partiziona i nodi di G.

[6] Spesso abbreviato con la sigla TSP, dall'inglese *traveling salesman problem*.

Per escludere le soluzioni in cui appaiono cicli parziali, possiamo aggiungere i seguenti vincoli di *subtour elimination*:

$$\sum_{e \in E[S]} x_e \leq |S| - 1, \quad \varnothing \neq S \subsetneq V,$$

dove $E[S]$ indica l'insieme degli archi che hanno entrambe le estremità in S. Per ogni S, questo vincolo vieta l'esistenza di un ciclo parziale il cui insieme dei nodi è S. Non è difficile verificare che, aggiungendo queste disequazioni, descriviamo esattamente l'insieme dei cicli hamiltoniani di G. (In realtà basterebbe restringersi ai sottoinsiemi S tali che $|S| \leq |V|/2$, perché se ci fosse un ciclo parziale che passa per più della metà dei nodi, i nodi non toccati da tale ciclo, che sono meno della metà del totale dei nodi, sarebbero a loro volta partizionati in cicli parziali. Tuttavia, per semplicità di notazione, scriviamo i vincoli per tutti i sottoinsiemi S.)

Otteniamo allora la seguente formulazione per il problema del commesso viaggiatore:

$$\min \quad \sum_{e \in E} c_e x_e \tag{2.76}$$

$$\text{s.a} \quad \sum_{e \in \delta^-(i)} x_e = 1, \qquad i \in V \tag{2.77}$$

$$\sum_{e \in \delta^+(i)} x_e = 1, \qquad i \in V \tag{2.78}$$

$$\sum_{e \in E[S]} x_e \leq |S| - 1, \quad \varnothing \neq S \subsetneq V \tag{2.79}$$

$$x_e \in \{0, 1\}, \qquad e \in E. \tag{2.80}$$

Il numero di vincoli di questo modello è estremamente grande, dunque anche solo risolvere il rilassamento lineare costituisce un problema dal punto di vista computazionale.

Prima di vedere come si può provare a superare questa difficoltà, diamo un modello equivalente a (2.76)–(2.80). Per ogni $\varnothing \neq S \subsetneq V$, la disequazione (2.79) può essere sostituita con

$$\sum_{e \in \delta^+(S)} x_e \geq 1, \tag{2.81}$$

dove $\delta^+(S)$ indica l'insieme degli archi che vanno da un nodo in S ad un nodo in $V \setminus S$. Infatti, poiché $\bigcup_{i \in S} \delta^+(i) = E[S] \cup \delta^+(S)$, sommando le equazioni (2.78) al variare di $i \in S$ otteniamo

$$\sum_{e \in E[S]} x_e + \sum_{e \in \delta^+(S)} x_e = |S|; \tag{2.82}$$

sottraendo (2.79) da quest'ultima equazione, ricaviamo (2.81). Viceversa, partendo da (2.81) e sottraendo l'equazione (2.82) si trova (2.79). Dunque un modello equivalente a (2.76)–(2.80) è il seguente (scriviamo già qui il rilassamento

lineare, perché di questo ci occuperemo nel seguito):

$$\min \quad \sum_{e \in E} c_e x_e \tag{2.83}$$

$$\text{s.a} \quad \sum_{e \in \delta^-(i)} x_e = 1, \quad i \in V \tag{2.84}$$

$$\sum_{e \in \delta^+(i)} x_e = 1, \quad i \in V \tag{2.85}$$

$$\sum_{e \in \delta^+(S)} x_e \geq 1, \quad \varnothing \neq S \subsetneq V \tag{2.86}$$

$$0 \leq x_e \leq 1, \qquad e \in E. \tag{2.87}$$

I vincoli (2.86) possono essere interpretati così: dato un sottoinsieme proprio S di nodi del grafo, affinché non ci sia un ciclo parziale su questi nodi, è necessario che almeno un arco della soluzione esca da S.

Generazione di righe per il rilassamento lineare

Per risolvere il programma lineare (2.83)–(2.87) ovviando alla presenza di un numero enorme di vincoli, cominciamo risolvendo il *problema ridotto*, ottenuto ignorando i vincoli (2.86). Sia $\bar{x}$ la soluzione ottima del problema ridotto. Vogliamo stabilire se $\bar{x}$ soddisfa tutte le disequazioni (2.86) (nel qual caso abbiamo terminato) oppure, in caso contrario, determinare un S per cui il corrispondente vincolo è violato. Si noti che questo obiettivo non può essere raggiunto semplicemente controllando i vincoli uno ad uno, perché il loro numero è troppo elevato.

Interpretando le componenti di $\bar{x}$ come capacità degli archi del grafo completo G, il problema di decidere se esiste $\varnothing \neq S \subsetneq V$ tale che $\sum_{e \in \delta^+(S)} \bar{x}_e < 1$ è il problema di decidere se esiste un *taglio* $\delta^+(S)$ di capacità minore di 1. Qui stiamo usando una nozione di taglio leggermente diversa da quella introdotta nella Sezione 2.5.4, dove avevamo fissato due nodi $s, t \in V$ e richiedevamo $s \in S$ e $t \notin S$. Ora chiamiamo taglio un qualunque insieme del tipo $\delta^+(S)$, dove $\varnothing \neq S \subsetneq V$. Per evidenziare la differenza tra le due definizioni, chiameremo i tagli discussi nella Sezione 2.5.4 "tagli che separano s da t".

Come trovare, se esiste, un taglio di capacità minore di 1? Per prima cosa, dimostriamo che, per ogni S, la capacità dei tagli $\delta^+(S)$ e $\delta^+(V \setminus S)$ è la stessa. Questo segue dall'equazione (2.82) e dall'analoga equazione che si ottiene sommando le equazioni (2.77):

$$\sum_{e \in E[S]} x_e + \sum_{e \in \delta^-(S)} x_e = |S|. \tag{2.88}$$

Confrontando (2.82) e (2.88), si trova $\sum_{e \in \delta^+(S)} x_e = \sum_{e \in \delta^-(S)} x_e$. Osservando che $\sum_{e \in \delta^-(S)} x_e = \sum_{e \in \delta^+(V \setminus S)} x_e$, si conclude che la capacità dei tagli $\delta^+(S)$ e $\delta^+(V \setminus S)$ è la stessa. Si noti che questa proprietà non è vera in generale, ma vale grazie alle particolari capacità che abbiamo assegnato agli archi.

Ora, poiché un taglio di capacità minore di 1 esiste se e solo se il taglio di capacità minima ha capacità minore di 1, quello che dobbiamo fare è determinare un taglio di capacità minima. Grazie al fatto che, per ogni S, la capacità dei tagli $\delta^+(S)$ e $\delta^+(V \setminus S)$ è la stessa, possiamo fissare un nodo $s \in V$ e cercare il taglio $\delta^+(S)$ che ha capacità minima, imponendo la condizione aggiuntiva $s \in S$. Nella Sezione 2.5.4 abbiamo visto che, per ogni coppia di nodi $s, t \in V$, è possibile determinare un taglio di capacità minima che separa s da t risolvendo un semplice programma lineare (ma ci sono anche algoritmi più specifici). Ogni taglio $\delta^+(S)$ con $s \in S$ separa necessariamente s da un qualche altro nodo $t \in V$. Poiché non conosciamo l'identità di t, cercheremo un taglio di capacità minima che separa s da t per tutte le possibili scelte di $t \in V \setminus \{s\}$.

Il procedimento appena descritto permette di determinare un taglio di capacità minima $\delta^+(S^*)$, dove $\varnothing \neq S^* \subsetneq V$. Ora, se la capacità di questo taglio è minore di 1, il vincolo (2.86) corrispondente al sottoinsieme S^* è violato dalla soluzione corrente $\bar{x}$. Possiamo dunque aggiungere questo vincolo al problema ridotto e riottimizzare (facendo uso, come di consueto, dell'algoritmo del simplesso duale). Se invece la capacità del taglio $\delta^+(S^*)$ è almeno 1, allora $\bar{x}$ non viola alcun vincolo (2.86) ed è dunque soluzione ottima del programma lineare (2.83)–(2.87).

Ricapitolando, l'algoritmo di generazione di righe per il rilassamento lineare del problema del commesso viaggiatore può essere così sintetizzato:

1. Risolviamo il problema ridotto, ottenuto da (2.83)–(2.87) ignorando i vincoli (2.86). Sia $\bar{x}$ una soluzione ottima di tale problema.

2. Determiniamo un taglio minimo $\delta^+(S^*)$ nel grafo completo in cui ad ogni arco e è assegnata capacità $\bar{x}_e$.

3. Se la capacità del taglio minimo è almeno 1, allora $\bar{x}$ è soluzione ottima di (2.83)–(2.87); altrimenti, aggiungiamo al problema ridotto il vincolo (2.86) corrispondente al sottoinsieme S^* e iteriamo.

Risolvere il programma lineare intero

La procedura di generazione di righe appena illustrata permette di risolvere il rilassamento lineare (2.83)–(2.87) senza dover considerare esplicitamente tutti i numerosissimi vincoli (2.86). A questo punto, per risolvere il programma lineare intero corrispondente, si può procedere con l'algoritmo di branch-and-bound. Poiché le variabili sono tutte binarie, ad ogni branching fisseremo il valore di una variabile a 0 o a 1. È facile verificare che il problema così ottenuto può essere visto come un nuovo problema del commesso viaggiatore su un grafo leggermente modificato (ad esempio assegnando costo molto alto o molto basso all'arco corrispondente alla variabile fissata). Dunque è possibile iterare il procedimento ed usare la tecnica di generazione di righe per risolvere i rilassamenti lineari dei sottoproblemi creati.

Un approccio alternativo consiste nell'effettuare la generazione di righe direttamente sul programma lineare intero. Possiamo partire risolvendo il problema ridotto in cui i vincoli (2.86) vengono ignorati, ma le variabili sono intere

anziché continue. Detta $\bar{x}$ la soluzione così ottenuta, verifichiamo se $\bar{x}$ individua un ciclo hamiltoniano o se invece contiene cicli parziali. Nel primo caso abbiamo terminato, mentre nel secondo caso aggiungiamo il vincolo (2.86) corrispondente ad un insieme di nodi S che individua un ciclo parziale e iteriamo. Anche se questa tecnica richiede di risolvere vari programmi lineari interi, questi avranno dimensioni contenute.

Sottolineiamo, comunque, il fatto che il problema del commesso viaggiatore è in generale difficile da risolvere e che le tecniche qui esposte sono più efficaci se combinate con altre metodologie non discusse in questo corso.

2.6.2 Schema generale

Come illustrato sopra, la tecnica della generazione di righe può essere usata per risolvere sia un programma lineare che un programma lineare intero (misto), ma in realtà l'ambito di applicazione è ben più vasto.

Supponiamo di avere un problema di ottimizzazione in cui i vincoli possono essere raggruppati in due famiglie V e W:

$$\begin{aligned} \min\ & f(x) \\ \text{s.a}\ & x \text{ soddisfa } v, \quad v \in V \\ & x \text{ soddisfa } w, \quad w \in W. \end{aligned}$$

Supponiamo che i vincoli della famiglia V siano trattabili (ad esempio perché sono pochi o perché esistono algoritmi efficienti per problemi con vincoli di questo tipo), ma che i vincoli della famiglia W, se considerati tutti assieme, non lo siano. Assumiamo anche di disporre di un algoritmo che, dato un vettore $\bar{x}$ che soddisfa tutti i vincoli in V, determini (se esiste) un vincolo in W violato da $\bar{x}$. Sotto tali ipotesi, è possibile applicare il metodo della generazione di righe:

1. Risolviamo il problema ridotto in cui i vincoli in W vengono ignorati, ottenendo una soluzione $\bar{x}$.
2. Determiniamo un vincolo $w^* \in W$ violato da $\bar{x}$ (se tale vincolo non esiste, $\bar{x}$ è soluzione ottima per il problema dato).
3. Aggiungiamo il vincolo w^* al problema ridotto e iteriamo.

2.7 Problemi con molte variabili: generazione di colonne

In questa sezione esaminiamo il caso simmetrico rispetto a quello della sezione precedente: problemi con un numero enorme di variabili. La tecnica che è possibile adottare sarà ilustrata per mezzo del problema di *cutting stock*.

2.7.1 Problema di cutting stock

Nel problema di cutting stock sono dati rotoli di lunghezza $L > 0$ ed una lista di tipologie di pezzi da tagliare, indicizzati con $i = 1, \ldots, n$. Per ogni tipologia

i, sono noti la lunghezza a_i (con $0 < a_i \leq L$) ed il numero di pezzi richiesti $b_i \in \mathbb{N}$. Si chiede di determinare come tagliare i pezzi richiesti dai rotoli, in modo da minimizzare il numero di rotoli utilizzati.

Formulazione naturale

Il modo più naturale di formulare il problema di cutting stock come programma lineare intero è probabilmente il seguente. Fissiamo un valore N sufficientemente grande, tale cioè che N rotoli siano certamente sufficienti a soddisfare la richiesta (ad esempio $N = \sum_{i=1}^{n} b_i$). Per ogni tipologia $i = 1, \ldots, n$ e rotolo $j = 1, \ldots, N$, definiamo una variabile intera x_{ij} che indica il numero di pezzi di tipo i ricavati dal rotolo j. Definiamo anche, per $j = 1 \ldots, N$, una variabile binaria y_j che vale 1 se e solo se il rotolo j viene utilizzato. Una possibile formulazione è allora la seguente:

$$\begin{aligned}
\min \ & \sum_{j=1}^{N} y_j & & \\
\text{s.a} \ & \sum_{j=1}^{N} x_{ij} \geq b_i, & & i = 1, \ldots, n \\
& \sum_{i=1}^{n} a_i x_{ij} \leq L y_j, & & j = 1, \ldots, N \\
& x_{ij} \in \mathbb{N}, \ y_j \in \{0, 1\}, & & i = 1, \ldots, n, \ j = 1, \ldots, N.
\end{aligned}$$

La prima famiglia di vincoli assicura che vengano tagliati tutti i pezzi richiesti, mentre il secondo gruppo di disequazioni svolge due compiti: fa sì che se da un rotolo viene ricavato qualche pezzo allora la corrispondente variabile y_j valga 1 e, in tal caso, impone che la lunghezza totale dei pezzi tagliati dal rotolo in questione non ecceda la lunghezza L.

Questo modello consta di un numero di vincoli e variabili molto elevato. Inoltre, l'esperienza mostra che questa formulazione è piuttosto debole: dopo aver risolto il rilassamento lineare (cosa già molto dispendiosa dal punto di vista computazionale), si è ancora molto lontani dall'ottimo intero.

Formulazione di Gilmore e Gomory

Una formulazione diversa, dovuta a Gilmore e Gomory (1961), si basa sul concetto di *schema di taglio* (o *pattern*). Uno schema indica in che modo usare un rotolo per ricavare alcuni dei pezzi richiesti. Formalmente, uno schema è un vettore $s \in \mathbb{N}^n$, in cui ogni componente s_i indica quanti pezzi di tipo i tagliare da un dato rotolo. Affinché lo schema sia compatibile con la lunghezza del rotolo, dovrà valere $\sum_{i=1}^{n} a_i s_i \leq L$. Indichiamo con $\mathcal{S}$ l'insieme di tutti i possibili schemi:

$$\mathcal{S} = \left\{ s \in \mathbb{N}^n : \sum_{i=1}^{n} a_i s_i \leq L \right\}. \tag{2.89}$$

Se definiamo, per ogni $s \in \mathcal{S}$, una variabile intera x_s che indica il numero di rotoli tagliati secondo lo schema s, possiamo scrivere la seguente formulazione per il problema di cutting stock:

$$\min \sum_{s\in\mathcal{S}} x_s \tag{2.90}$$

$$\text{s.a} \sum_{s\in\mathcal{S}} s_i x_s \geq b_i, \qquad i = 1,\ldots,n \tag{2.91}$$

$$x_s \geq 0,\ x_s \in \mathbb{Z}, \quad s \in \mathcal{S}. \tag{2.92}$$

Questa formulazione, che consta di pochi vincoli ma moltissime variabili, presenta due grossi vantaggi rispetto alla precedente: (i) il rilassamento lineare può essere risolto senza dover considerare esplicitamente tutte le variabili; (ii) risolvere il rilassamento lineare è sufficiente ad ottenere una soluzione intera approssimata che, nella maggior parte dei casi, può essere considerata soddisfacente. Cominciamo con l'illustare quest'ultimo punto, rinviando il primo a più sotto.

Nella seguente proposizione, data una soluzione ammissibile x per il rilassamento lineare di (2.90)–(2.92), indichiamo con $z(x)$ il corrispondente valore della funzione obiettivo: $z(x) = \sum_{s\in\mathcal{S}} x_s$. Il seguente risultato mostra che arrotondando per eccesso tutte le componenti di una soluzione ottima di base del rilassamento lineare, si ottiene una "buona" soluzione intera.

Proposizione 2.17 *Con riferimento alla forma standard del problema* (2.90)–(2.92)*, sia $\bar{x}$ una soluzione ottima di base del rilassamento lineare e sia x^* una soluzione ottima intera. Allora il vettore $\lceil \bar{x} \rceil$ è una soluzione ammissibile che soddisfa $0 \leq z(\lceil \bar{x} \rceil) - z(x^*) \leq n - 1$.*

Dimostrazione. L'ammissibilità di $\lceil \bar{x} \rceil$ è ovvia. Inoltre, poiché x^* è soluzione ottima intera, deve valere $z(x^*) \leq z(\lceil \bar{x} \rceil)$. Resta da verificare che $z(\lceil \bar{x} \rceil) - z(x^*) \leq n - 1$. Poiché il problema (2.90)–(2.92), messo in forma standard, è definito da un sistema $Ax = b$ in cui A ha n righe, ogni base contiene esattamente n elementi. Ne segue che $\bar{x}$ ha al massimo n componenti diverse da zero. Da questo e dal fatto che $z(\bar{x}) \leq z(x^*)$ si deduce che

$$z(\lceil \bar{x} \rceil) - z(x^*) \leq z(\lceil \bar{x} \rceil) - z(\bar{x}) = \sum_{s\in S} (\lceil \bar{x}_s \rceil - \bar{x}_s) < n,$$

dove abbiamo usato il fatto che al massimo n termini dell'ultima sommatoria sono diversi da zero (e sono comunque minori di 1). Poiché il primo membro è un numero intero, otteniamo $z(\lceil \bar{x} \rceil) - z(x^*) \leq n - 1$. □

Generazione di colonne

Per risolvere il rilassamento lineare del problema (2.90)–(2.92) senza considerare esplicitamente tutti gli schemi $s \in \mathcal{S}$, è possibile ricorrere ad una tecnica nota come *generazione di colonne*. Come vedremo, questa procedura è, in sostanza, la generazione di righe applicata al problema duale.

Cominciamo col definire un *problema ridotto*, in cui solo un sottoinsieme di schemi $\hat{\mathcal{S}} \subseteq \mathcal{S}$ (e dunque solo un sottoinsieme di variabili) viene considerato:

$$\min \sum_{s \in \hat{\mathcal{S}}} x_s \tag{2.93}$$

$$\text{s.a} \sum_{s \in \hat{\mathcal{S}}} s_i x_s \geq b_i, \quad i = 1, \ldots, n \tag{2.94}$$

$$x_s \geq 0, \qquad s \in \hat{\mathcal{S}}. \tag{2.95}$$

Vogliamo determinare $\hat{\mathcal{S}}$ in modo che il programma lineare (2.93)–(2.95) abbia almeno una soluzione ammissibile. È chiaro che questa condizione sarà certamente verificata se, per ogni $i = 1, \ldots, n$, includiamo in $\hat{\mathcal{S}}$ uno schema $s^{(i)}$ che preveda di ricavare solo pezzi di tipo i. Formalmente, possiamo prendere $s^{(i)} = \lfloor L/a_i \rfloor e_i$ per ogni i. Il problema ridotto così ottenuto ha dimensioni contenute e può essere risolto facilmente.

Applicando il metodo del simplesso, otteniamo una coppia di soluzioni ottime $\bar{x}, \bar{u}$ per il problema ridotto ed il suo duale

$$\max \sum_{i=1}^{n} b_i u_i \tag{2.96}$$

$$\text{s.a} \sum_{i=1}^{n} s_i u_i \leq 1, \quad s \in \hat{\mathcal{S}} \tag{2.97}$$

$$u_i \geq 0, \qquad i = 1, \ldots, n. \tag{2.98}$$

Si noti che, sebbene $\bar{x}$ sia un vettore con componenti associate ai soli schemi $s \in \hat{\mathcal{S}}$, possiamo estendere $\bar{x}$ ponendo a zero le componenti associare agli altri schemi, ottenendo così una soluzione ammissibile per il *problema master* (così è chiamato il rilassamento lineare di (2.90)–(2.92), cioè il programma lineare in cui tutti gli schemi sono considerati).

Vogliamo ora stabilire se $\bar{x}$ e $\bar{u}$ sono soluzioni ottime anche per il problema master ed il suo duale. Osserviamo, prima di tutto, che il duale del problema master è esattamente (2.96)–(2.98), con la sola differenza che i vincoli (2.97) vanno scritti per ogni $s \in \mathcal{S}$ invece che per $s \in \hat{\mathcal{S}}$. Ora, per il Corollario 1.4, per decidere se $\bar{x}$ e $\bar{u}$ sono soluzioni ottime anche per il problema master ed il suo duale, basterà verificare se $\bar{x}$ e $\bar{u}$ sono ammissibili per i rispettivi problemi e se $\sum_{s \in \mathcal{S}} \bar{x}_s = \sum_{i=1}^{n} b_i \bar{u}_i$.

L'ammissibilità primale di $\bar{x}$ è soddisfatta, come già osservato. Inoltre, poiché $\bar{x}$ e $\bar{u}$ sono soluzioni ottime per il problema ridotto ed il suo duale, per il Teorema di Dualità Forte si ha $\sum_{s \in \hat{\mathcal{S}}} \bar{x}_s = \sum_{i=1}^{n} b_i \bar{u}_i$, che implica $\sum_{s \in \mathcal{S}} \bar{x}_s = \sum_{i=1}^{n} b_i \bar{u}_i$. Dunque l'unica cosa da verificare è se $\bar{u}$ sia ammissibile per il duale del problema master, cioè se

$$\sum_{i=1}^{n} s_i \bar{u}_i \leq 1, \quad s \in \mathcal{S}. \tag{2.99}$$

Questa proprietà vale se e solo se

$$\max \left\{ \sum_{i=1}^{n} s_i \bar{u}_i : s \in \mathcal{S} \right\} \leq 1.$$

Ricordando la definizione di $\mathcal{S}$ data in (2.89), quest'ultima condizione può essere parafrasata dicendo che il valore ottimo del seguente programma lineare deve essere non superiore a 1:

$$\max \sum_{i=1}^{n} \bar{u}_i s_i \tag{2.100}$$

$$\text{s.a} \sum_{i=1}^{n} a_i s_i \leq L \tag{2.101}$$

$$s_i \geq 0,\ s_i \in \mathbb{Z}, \quad i = 1, \ldots, n. \tag{2.102}$$

Questo programma lineare intero, detto *problema di pricing*, è una variante del problema dello zaino (Sezione 2.2.1) in cui sono disponibili più copie di ciascun oggetto. Date le sue dimensioni contenute, questo problema può essere risolto in modo abbastanza efficiente.

Ci sono ora due possibilità. Se il valore ottimo del problema di pricing non eccede 1, allora, per quanto detto, $\bar{x}$ è una soluzione ottima per il problema master e abbiamo terminato. Altrimenti, detta s^* una soluzione ottima del problema di pricing, $\bar{u}$ viola il vincolo (2.99) corrispondente allo schema s^*. Possiamo allora aggiungere s^* a $\hat{\mathcal{S}}$, ingrandendo così il problema ridotto, e iterare.

Ricapitolando, l'algoritmo di generazione di colonne può essere così schematizzato:

1. Costruiamo un "piccolo" sottoinsieme $\hat{\mathcal{S}} \subseteq \mathcal{S}$ in modo tale che il problema ridotto abbia una soluzione ammissibile.

2. Risolviamo il problema ridotto (2.93)–(2.95), ottenendo una coppia di soluzioni ottime primale e duale $\bar{x}, \bar{u}$.

3. Risolviamo il problema di pricing (2.100)–(2.102), ottenendo una soluzione ottima s^*.

4. Se $\sum_{i=1}^{n} \bar{u}_i s_i^* \leq 1$, allora $\bar{x}$ è soluzione ottima del problema master ed abbiamo terminato. Altrimenti, aggiungiamo s^* a $\hat{\mathcal{S}}$ e torniamo al passo 2.

2.7.2 Schema generale

In generale, dato un programma lineare con moltissime variabili e pochi vincoli, si può tentare di applicare la tecnica di generazione di colonne. L'algoritmo è sostanzialmente quello illustrato sopra per il problema di cutting stock, con l'unica differenza che il problema di pricing avrà una forma diversa a seconda del problema di partenza. Vediamo brevemente come funziona la tecnica in generale e quali condizioni devono valere affinché la si possa utilizzare.

Una prima condizione che deve essere soddisfatta è la seguente: dobbiamo essere in grado di determinare un piccolo sottoinsieme di variabili (cioè colonne della matrice dei vincoli) tale che, restringendo il problema a queste sole colonne, ci sia una soluzione ammissibile.

Costruito un tale problema ridotto, possiamo determinare una coppia di soluzioni ottime $\bar{x}, \bar{u}$ per il problema ridotto ed il suo duale. La soluzione primale $\bar{x}$ si estende in modo ovvio ad una soluzione ammissibile del problema master. Per decidere se $\bar{x}$ e $\bar{u}$ sono ottime anche per il problema master ed il suo duale, si deve verificare se $\bar{u}$ soddisfa tutti i vincoli del duale del problema master. Poiché i vincoli del duale sono numerosissimi, questo obiettivo non può essere raggiunto semplicemente controllando tutti i vincoli uno ad uno; al contrario, dobbiamo disporre di un algoritmo che determini in modo efficiente se tutti i vincoli duali sono soddisfatti, o, in caso contrario, ne trovi uno violato da $\bar{u}$.[7] Il problema di trovare un vincolo duale violato è detto problema di pricing. Nel caso del cutting stock si ottiene un problema dello zaino, ma più in generale la forma del problema di pricing dipende dal problema di partenza.

Se non esistono vincoli duali violati, allora la coppia $\bar{x}, \bar{u}$ è ottima anche per il problema master ed il suo duale; altrimenti, dato un vincolo duale violato, aggiungiamo al problema ridotto la colonna corrispondente a questo vincolo duale e iteriamo.

Sebbene non ci sia garanzia che un tale algoritmo termini in un numero ragionevole di iterazioni, c'è un buon motivo per essere ottimisti: poiché il problema master contiene pochi vincoli, in una soluzione (ottima) di base ci saranno poche componenti diverse da zero. Ciò significa che solo poche colonne sono necessarie per trovare l'ottimo. Questo autorizza a sperare che tutte le colonne necessarie vengano trovate in un numero ragionevole di iterazioni.

Infine, nel caso in cui il problema in esame sia un programma lineare intero (misto) con moltissime variabili, è possibile fare ricorso ad una strategia chiamata *branch-and-price*: il branch-and-price combina branch-and-bound e generazione di colonne, utilizzando quest'ultima tecnica per risolvere i rilassamenti lineari dei vari sottoproblemi creati. Quando a tutto questo si aggiunge anche l'utilizzo di piani di taglio, si parla di *branch-and-cut-and-price*.

[7] Si noti che questa è di fatto la generazione di righe per il problema duale.