Università degli Studi di Padova

Facoltà di ingegneria

Dipartimento di tecnica e gestione dei sistemi industriali

Tesi di laurea di primo livello

OTTIMIZZAZIONE INTERA

Laureando: Francesca Pippa

Relatore: prof. Giorgio Romanin Jacur

Anno accademico 2010-2011

INDICE

INTRODUZIONE

La ricerca operativa è una scienza che fornisce strumenti matematici di supporto alle attività decisionali in cui occorre gestire e coordinare attività e risorse limitate al fine di massimizzare o minimizzare una funzione obiettivo. Si tratta di formalizzare un problema in un modello matematico e calcolare una soluzione ottima, quando possibile, o approssimata per esso. Esso costituisce un approccio scientifico alla risoluzione di problemi complessi, si può ricondurre all'ambito della matematica, informatica, economia e finanza, ingegneria ed altre. Inoltre la ricerca operativa ha numerose applicazioni commerciali soprattutto nell'ambito economico, infrastrutturale, logistico, militare, della progettazione di servizi e sistemi di trasporto e nelle tecnologie. Riveste un ruolo importante nelle attività decisionali perchè permette di operare le scelte migliori per raggiungere un determinato obiettivo rispettando vincoli che sono imposti dall'esterno e non sono sotto il controllo di chi deve compiere le decisioni.

La programmazione lineare (PL), in particolare, è una branca della ricerca operativa che si occupa di studiare algoritmi di risoluzione per problemi di ottimizzazione lineari. Un problema è detto lineare se sia la funzione oggetto che le funzioni che definiscono i vincoli sono lineari. Un problema di programmazione lineare consiste quindi nel ricercare i valori delle variabili decisionali che massimizzano (o minimizzano) una funzione lineare e che verificano un insieme di disuguaglianze lineari.

Tra i vari modelli della materia esistenti, questo è quello che viene più ampiamente utilizzato, infatti non solo si applica a numerosi problemi reali, ma è anche un importante strumento di supporto nell'analisi e nella risoluzione di problemi di programmazione matematica più complessi.

Tanti modelli di PL sono continui, nel senso che le variabili di decisione possono essere frazionarie. Spesso questa è un'assunzione realistica. Per esempio, possiamo facilmente produrre 102 e ¾ galloni di un bene divisibile come il vino. Altre volte, invece, le soluzioni frazionarie non sono realistiche, e dobbiamo far assumere alle variabili decisionali valori interi. In molti casi è però possibile eliminare la condizione di interezza, in quanto risultano trascurabili gli eventuali arrotondamenti della soluzione ottimale.

Esistono tuttavia altri modelli di ottimizzazione per i quali la condizione di interezza diventa irrinunciabile. Si tratta, per esempio, dei modelli di ottimizzazione binaria, nei quali le variabili assumono i valori 0 e 1, con il significato di 'vero' e 'falso' riferito ad alcune condizioni logiche. Per tali sistemi, non sono ancora stati scoperti degli algoritmi polinomiali per la loro

risoluzione; le tecniche utilizzate sono molteplici, e comprendono il metodo dei *piani di taglio* e il metodo di *branch and bound*, che analizzeremo in dettaglio.

Ho scelto di sviluppare nella mia tesi questo argomento perché è molto incentrato sui problemi reali, che si ritrovano tutti i giorni negli ambienti lavorativi comuni. Ritengo sia molto importante che la teoria di un argomento studiato possa trovare riscontro nella realtà, e so inoltre che questo è un campo in cui si scopriranno sempre nuovi modi per risolvere al meglio i diversi problemi di ottimizzazione.

Nel primo capitolo, analizzerò nel dettaglio i vari modelli di programmazione lineare intera che si possono utilizzare per risolvere diversi tipi di problematiche reali, sia introducendoli dal punto di vista teorico che chiarendoli attraverso un esempio numerico.
Nel secondo capitolo, spiegherò brevemente le proprietà fondamentali della programmazione intera, anche attraverso l'utilizzo di grafici.
Nel terzo ed ultimo capitolo, infine, esporrò le principali tecniche di risoluzione, spiegandolo anche attraverso la risoluzione pratica di diversi problemi.

CAPITOLO 1
Esempi di problemi di ottimizzazione lineare intera

In linea generale, un problema di ottimizzazione lineare intera mista è formulato come:

max $\mathbf{c'x + d'y}$
s.a $\mathbf{Ax + Ey = b}$
$\mathbf{x \in Z^n_+}$, $\mathbf{y \geq 0}$

dove:

- **x** è un vettore di n variabili di decisione che possono assumere valori appartenenti agli interi non negativi $\mathbf{Z}^n_+$;
- $\mathbf{y \in R}^q$ è un vettore di variabili continue;
- $\mathbf{c \in R}^n$, $\mathbf{d \in R}^q$ sono i corrispondenti vettori dei coefficienti di profitto;
- $\mathbf{b \in R}^m$ è il vettore dei termini noti;
- $\mathbf{A \in R}^{mxn}$, $\mathbf{E \in R}^{mxq}$ sono le matrici dei vincoli.

Nel caso in cui la condizione di interezza sia riferita a tutte le variabili del problema, e quindi si abbia $q = 0$, il problema viene definito di ottimizzazione lineare intera pura. Se la condizione di interezza viene sostituita dalla richiesta che le variabili assumano solo i valori 0 e 1, ovvero $\mathbf{x \in} \{0,1\}^n$, abbiamo un problema di ottimizzazione binaria o booleana.

1.Pianificazione produttiva con lotti minimi

Per analizzare il modello di pianificazione produttiva con lotti minimi dobbiamo fare una digressione andando a studiare innanzitutto la pianificazione produttiva multi-periodo. Questa a sua volta è un'estensione del problema di mix produttivo.

1.1.Mix produttivo

Consideriamo dunque il seguente problema di pianificazione produttiva. Sono dati n prodotti, indicati come $j = 1, 2,..., n$, che competono per l'utilizzo di m risorse, indicate come $i = 1, 2,..., m$, la cui disponibilità totale è limitata, necessarie al compimento del processo di trasformazione produttiva. Ogni prodotto j è caratterizzato da un margine lordo unitario c_j e

da un assorbimento unitario della risorsa *i* pari a a_{ij}. A ciascuna risorsa *i* è associata una disponibilità massima b_i, assegnata sull'orizzonte temporale di pianificazione.
Il problema di mix produttivo richiede di determinare un piano di produzione che massimizzi il margine lordo nel rispetto dei vincoli di capacità delle risorse. Se indichiamo con x_j la quantità di prodotto *j* che dev'essere realizzata, possiamo formulare il seguente modello di ottimizzazione, avente *n* variabili di decisione e *m* vincoli

$$
\begin{aligned}
\max \quad & c_1x_1 + c_2x_2 + \ldots + c_nx_n \\
s.a \quad & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \\
& a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2 \\
& \ldots \\
& a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m \\
& x_1, x_2, \ldots, x_n \geq 0
\end{aligned}
$$

Tale modello richiede di massimizzare il margine lordo, rispetto alle variabili di decisione x_j, in presenza di *m* vincoli di capacità. Il vincolo *i*-esimo impone che la quantità di risorsa *i* utilizzata, rappresentata dal primo membro della corrispondente disequazione, sia non superiore alla disponibilità b_i che figura a secondo membro. Occorre inoltre imporre le condizioni di non negatività per le variabili di decisione.
Facciamo un esempio pratico, considerando il caso di mix produttivo per un'azienda alimentare.
Un'azienda alimentare produce due diversi formati di pasta, farfalle e rigatoni, e desidera ricavare il piano ottimale di produzione per un'orizzonte temporale costituito da un unico periodo, pari a una settimana. Il ciclo produttivo prevede tre fasi critiche, corrispondenti alle risorse limitate che devono essere allocate in modo ottimale ai due prodotti: la macinazione delle semole, la produzione e la confezione della pasta. Per ogni quintale del formato farfalle viene utilizzata 1 ora per la fase di macinazione, 4 per la produzione e 2 per la confezione. Analogamente, l'assorbimento di capacità per realizzare un quintale del formato rigatoni è paro a 1 ora per la macinazione, e per la produzione e e per la confezione. La disponibilità di capacità produttiva nell'orizzonte temporale è pari a 40 ore per la macinazione, 132 ore per la produzione e 140 ore per la confezione. Il margine lordo unitario è pari a 24 euro per quintale per le farfalle e 18 euro per quintale per i rigatoni. I dati del problema sono riassunti nella seguente tabella

		assorbimento	unitario
	capacità (h)	farfalle	rigatoni
macinazione	40	1	1
produzione	132	4	2
confezione	140	2	4
	margine (€/q)	24	18

Si suppone inoltre che, entro i limiti della capacità produttiva disponibile, l'azienda possa collocare sul mercato la quantità prodotta per ciascun formato. L'obiettivo dell'azienda consiste nel determinare un piano di produzione, rappresentato da un mix dei due prodotti, tale da massimizzare il margine lordo totale, nel rispetto dei limiti di capacità delle risorse disponibili.

Per formulare un modello di ottimizzazione che rappresenti il processo decisionale descritto, è necessario in primo luogo identificare e rappresentare simbolicamente le variabili di decisione. In questo caso le decisioni corrispondono al volume di produzione per ciascuno dei due formati di pasta. Per indicare le variabili decisionali utilizziamo i simboli F e R, con il seguente significato:

F = quintali di pasta del formato farfalle da produrre

R = quintali di pasta del formato rigatoni da produrre.

Il passo successivo consiste nell'esprimete la funzione obiettivo del modello. Indicando come z il margine lordo corrispondente al mix produttivo (F, R), otteniamo la relazione

margine lordo: $z = 24\,F + 18\,R$.

Infine, per completare la formulazione del modello di ottimizzazione, è necessario esprimere i vincoli di capacità produttiva mediante disequazioni relative alle variabili di decisione. Consideriamo dapprima la fase di macinazione, la cui disponibilità è pari a 40 ore. Possiamo supporre che la realizzazione del mix produttivo (F, R) comporti un assorbimento proporzionale e additivo, ovvero lineare, della capacità di macinazione, formulando la relazione

capacità di macinazione: $F + R \leq 40$.

Essa impone che la capacità assorbita, espressa dal primo membro della disequazione, sia non superiore alla capacità disponibile, a sua volta rappresentata dal secondo membro. Applicando un analogo ragionamento alle fasi si produzione e confezione, otteniamo rispettivamente le disequazioni

capacità di produzione: $4F + 2R \leq 132$

capacità di confezione: $2F + 4R \leq 140$.

Da ultimo, è necessario introdurre nel modello due vincoli per esprimere le condizioni di non negatività per le variabili di decisione

vincoli di non negatività: F ≥ 0, R ≥ 0.

Riassumendo, il problema di mix ottimale di produzione è riconducibile al seguente modello di ottimizzazione lineare

$$\begin{aligned} \max \quad & 24F + 18R \\ s.a \quad & F + R \leq 40 \\ & 4F + 2R \leq 132 \\ & 2F + 4R \leq 140 \\ & F, R \geq 0. \end{aligned}$$

1.2.Pianificazione produttiva multi-periodo

Consideriamo ora il problema di mix produttivo e supponiamo di sviluppare una pianificazione multi-periodo. L'orizzonte temporale è suddiviso in *T* intervalli equiampi, che di solito corrispondono alle settimane o ai mesi, e che indicheremo con l'indice *t* = 1, 2,..., *T*. Per ogni periodo è assegnata una domanda di prodotti che dev'essere soddisfatta. E' tuttavia possibile accumulare scorte di prodotti realizzate in un periodo e utilizzabili per soddisfare la domanda in quello successivo. In questo modo il sistema può compensare le fluttuazioni nei livelli della domanda utilizzando con efficienza la capacità produttiva disponibile.

Il modello di pianificazione multi-periodo prevede di determinare il volume produttivo per i *T* periodi dell'orizzonte temporale, in modo da soddisfare la domanda e i vincoli di capacità, minimizzando i costi variabili di produzione e i costi delle scorte. Le variabili di decisione sono

P_{jt} = unità di prodotto *j* da realizzare nel periodo *t*

I_{jt} = unità di prodotto *j* a scorta alla fine del periodo *t*,

mentre i parametri hanno il seguente significato

d_{jt} = domanda del prodotto *j* nel periodo *t,*

c_{jt} = costo unitario di produzione del prodotto *j* nel periodo *t,*

h_{jt} = *costo unitario di mantenimento a scorta del prodotto j nel periodo t,*

a_{jt} = assorbimento della risorsa *i* per realizzare una unità di prodotto *j,*

b_{jt} = *capacità* della risorsa *i* disponibile nel periodo *t*.

Il problema di pianificazione produttiva multi-periodo è formulato mediante il seguente modello di ottimizzazione lineare

$$\min \sum_{t=1}^{T} \sum_{j=1}^{n} (c_{jt}P_{jt} + h_{jt}I_{jt})$$

$$s.a \quad P_{jt} + I_{j,t-1} - I_{jt} = d_{jt} \qquad j = 1,2,...,n, \quad t = 1,2,...,T$$

$$\sum_{j=1}^{n} a_{ij}P_{jt} \leq b_{it} \qquad i = 1,2,...,m, \quad t = 1,2,...,T$$

$$P_{jt}, I_{jt} \geq 0 \qquad j = 1,2,...,n, \; t = 1,2,...,T.$$

La funzione obiettivo tiene conto dei costi di produzione e dei costi di mantenimento a scorta. I primi vincoli assicurano, per ogni prodotto *j* e per ogni periodo *t*, che la quantità di prodotto realizzata nel periodo più la scorta del precedente periodo sottratta alla scorta di quello attuale, corrisponda alla domanda del prodotto *j* nel periodo *t*. I secondi impongono che la quantità di ciascuna risorsa *i* assorbita in ciascun periodo *t* non superi la capacità disponibile.

Facciamo ora un esempio numerico, programmando la pianificazione produttiva per un'azienda alimentare.

Consideriamo un'estensione multi-periodo dell'esempio precedente. Supponiamo che la pianificazione vada effettuata su tre periodi, indicati come *t* = 1, 2, 3. La capacità produttiva per le tre fasi di macinazione, produzione e confezione è pari a 40, 132, 140, rispettivamente, in ognuno dei tre periodi considerati. Gli assorbimenti unitari di capacità produttiva e i costi di produzione sono a loro volta costanti nei tre periodi, pari ai valori considerati nell'esempio qui sopra, con i costi uguali ai profitti. I costi unitari di mantenimento a scorta sono pari a 0.2 e 0.09 euro per quintale, rispettivamente per il formato farfalle e rigatoni, e sono costanti per i tre periodi. La domanda dei due prodotti nei tre periodi è descritta dalla seguente tabella

	domanda (q)	
periodo	farfalle	rigatoni
1	16	16
2	28	14
3	28	18

La formulazione del problema di pianificazione multi-periodo, corrispondente al modello di ottimizzazione lineare riportato nella pagina precedente, è questa

$$\min \quad 24F_1 + 18R_1 + 24F_2 + 18R_2 + 24F_3 + 18R_3 + 0,2I_{F1} + 0.09I_{R1} + 0,2I_{F2} + 0.09I_{R2}$$

$$s.a \quad F_1 + R_1 \leq 40$$
$$4F_1 + 2R_1 \leq 132$$
$$2F_1 + 4R_1 \leq 140$$
$$F_2 + R_2 \leq 40$$
$$4F_2 + 2R_2 \leq 132$$
$$2F_2 + 4R_2 \leq 140$$
$$F_3 + R_3 \leq 40$$
$$4F_3 + 2R_3 \leq 132$$
$$2F_3 + 4R_3 \leq 140$$
$$F_1 - I_{F1} = 16$$
$$R_1 - I_{R1} = 16$$
$$F_2 + I_{F1} - I_{F2} = 28$$
$$R_2 + I_{R1} - I_{R2} = 14$$
$$F_3 + I_{F2} = 28$$
$$R_3 + I_{R2} = 28$$
$$F_1, R_1, F_2, R_2, F_3, R_3, I_{F1}, I_{R1}, I_{F2}, I_{R2} \geq 0.$$

1.3.Pianificazione produttiva con lotti minimi

Consideriamo il modello di pianificazione produttiva multi-periodo e supponiamo che per alcuni prodotti debba essere rispettata una condizione di lotto minimo. Infatti, per motivi tecnologici o per ragioni di convenienza economica è spesso necessario imporre che il volume produttivo di uno o più prodotti sia alternativamente pari a 0 oppure maggiore di una soglia specificata, detta appunto lotto minimo.

Per rappresentare condizioni di lotto minimo è necessario introdurre, accanto alle variabili di produzione P_{jt}, le seguenti variabili binarie

Y_{jt} = 1 se $P_{jt} > 0$

0 altrimenti

e i parametri

l_j = dimensione minima del lotto di prodotto j,

α = costante maggiore del volume producibile per j.

Possiamo formulare il modello di ottimizzazione intera mista

$$\min \quad \sum_{t=1}^{T} \sum_{j=1}^{n} (c_{jt}P_{jt} + h_{jt}I_{jt})$$

$$s.a \quad P_{jt} + I_{j,t-1} - I_{jt} = d_{jt} \qquad j = 1,2,...,n, \quad t = 1,2,...,T$$

$$\sum_{j=1}^{n} a_{ij}P_{jt} \leq b_{it} \qquad i = 1,2,...,m, \quad t = 1,2,...,T$$

$$P_{jt} \geq l_j Y_{jt} \qquad j = 1,2,...,n, \quad t = 1,2,...,T$$
$$P_{jt} \leq \alpha\, Y_{jt} \qquad j = 1,2,...,n, \quad t = 1,2,...,T$$
$$P_{jt}, I_{jt} \geq 0 \quad Y_{jt} \in \{0,1\} \qquad j = 1,2,...,n, \quad t = 1,2,...,T$$

I penultimi vincoli esprimono le condizioni di lotto minimo. Ciascuno degli ultimi vincoli rappresenta invece una condizione di conseguenza logica tre la variabili Y_{jt} e P_{jt} per forzare al valore 1 la variabile binaria Y_{jt} allorchè il corrispondente volume produttivo P_{jt} è maggiore di 0. La costante α deve essere scelta sufficientemente grande da non costituire un'effettiva limitazione alla quantità realizzabile di prodotto *j*. Quest'ultima deve infatti essere limitata soltanto alla capacità disponibile e dalla domanda assegnata.
Si osservi che i penultimi vincoli non sono sufficienti a garantire che le condizioni di lotto minimo siano soddisfatte. Essi non assicurano infatti che le variabili binarie assumano il valore 1 in corrispondenza di valori positivi delle variabili di produzione associate. Soltanto l'effetto combinato di entrambi i vincoli permette di imporre le condizioni di lotto minimo.

Facciamo un esempio numerico.
Consideriamo il problema di pianificazione multi-periodo per un'azienda alimentare e supponiamo che il lotto minimo di produzione sia pari a 23 quintali per il formato farfalle e a 18 quintali per i rigatoni. Per rappresentare le condizioni di lotto minimo è necessario introdurre nel modello una variabile binaria per ogni prodotto e per ogni periodo, per un totale di 6 nuove variabili, indicate come Y_{F1}, Y_{F2}, Y_{F3}, Y_{R1}, Y_{R2}, e Y_{R3}. Per imporre il rispetto delle condizioni di lotto minimo occorre aggiungere al modello formulato precedentemente i seguenti vincoli, corrispondenti alle ultime e penultime relazioni del modello di ottimizzazione lineare intera mista per il problema di pianificazione produttiva con lotti minimi

$$F_1 \geq 23Y_{F1}$$
$$F_2 \geq 23Y_{F2}$$
$$F_3 \geq 23Y_{F3}$$
$$R_1 \geq 18Y_{R1}$$
$$R_2 \geq 18Y_{R2}$$
$$R_3 \geq 18Y_{R3}$$
$$F_1 \leq 120Y_{F1}$$
$$F_2 \leq 120Y_{F2}$$
$$F_3 \leq 120Y_{F3}$$
$$R_1 \leq 120Y_{R1}$$
$$R_2 \leq 120Y_{R2}$$
$$R_3 \leq 120Y_{R3}.$$

La costante α = 120 è stata calcolata come domanda complessiva dei due prodotti nei tre periodi dell'orizzonte di pianificazione.

2.Pianificazione produttiva con costi fissi

Nella formulazione dei modelli di mix produttivo e di pianificazione produttiva multi-periodo abbiamo implicitamente ipotizzato che i costi di produzione siano proporzionali al volume prodotto. Questa assunzione si è tradotta nella forma lineare **c'x** della funzione obiettivo.
Per alcuni sistemi logistici, la realizzazione di un prodotto richiede un costo fisso di attrezzaggio, indipendentemente dal volume produttivo pianificat0, che deve essere sostenuto soltanto se il volume produttivo è strettamente maggiore di 0, ovvero se la produzione del corrispondente prodotto viene attivata.
Più precisamente, supponiamo che in un problema di pianificazione produttiva per ogni prodotto *j* sia assegnato il nuovo parametro

f_{jt} = costo unitario di attrezzaggio per ilo prodotto *j* nel periodo *t*.

Se indichiamo con P_{jt} le variabili che esprimono il volume dei diversi prodotti nel periodo *t* per il problema di pianificazione produttiva, la funzione di costo per il *j*-esimo prodotto può essere espressa come

$$C_j(P_{jt}) = f_{jt} + c_{jt}P_{jt} \quad \text{se } P_{jt} > 0$$
$$0 \quad \text{altrimenti.}$$

La funzione di costo $C_j(P_{jt})$ è non lineare e presenta un punto di discontinuità nell'origine. Pertanto, il modello di pianificazione produttiva con costi fissi risulta a sua volta un problema di ottimizzazione non lineare e non differenziabile, formulato come

$$\min \quad \sum_{t=1}^{T}\sum_{j=1}^{n}(C_j(P_{jt}) + h_{jt}I_{jt})$$
$$s.a \quad P_{jt} + I_{j,t-1} - I_{jt} = d_{jt} \qquad j = 1,2,...,n, \quad t = 1,2,...,T$$
$$\sum_{j=1}^{n} a_{ij}P_{jt} \le b_{it} \qquad i = 1,2,...,m, \quad t = 1,2,...,T$$
$$P_{jt}, I_{jt} \ge 0 \qquad j = 1,2,...,n, \quad t = 1,2,...,T.$$

E' tuttavia possibile trasformare il problema di ottimizzazione non lineare in un modello di ottimizzazione binaria. Introduciamo una variabile binaria per ogni prodotto, in analogia con quanto visto per il modello con lotti minimi, ponendo

Y_{jt} = 1 se $P_{jt} > 0$
0 altrimenti.

Il problema di pianificazione produttiva con costi fissi può quindi essere formulato come modello di ottimizzazione lineare intera mista

$$\min \sum_{t=1}^{T} \sum_{j=1}^{n} (c_{jt}P_{jt} + h_{jt}I_{jt} + f_{jt}Y_{jt})$$

$$s.a \quad P_{jt} + I_{j,t-1} - I_{jt} = d_{jt} \qquad \forall j, \forall t$$

$$\sum_{j=1}^{n} a_{ij}P_{jt} \leq b_{it} \qquad \forall i, \forall t$$

$$P_{jt} \leq \alpha Y_{jt} \qquad \forall j, \forall t$$

$$P_{jt}, I_{jt} \geq 0, \quad Y_{jt} \in \{0,1\} \qquad \forall j, \forall t.$$

L'ultimo vincolo esprime la condizione di consistenza logica tra le variabili Y_{jt} e P_{jt}. Il parametro α costituisce una limitazione superiore alla quantità di prodotto *j* che può essere realizzata, e viene utilizzato nel vincolo di forzatura al valore 1 delle variabili binarie, come già per il modello di pianificazione produttiva con lotti minimi. Si osservi che la corrispondente forzatura delle variabili binarie Y_{jt} al valore 0 avviene per effetto della loro presenza nella funzione obiettivo con un coefficiente positivo. E' facile verificare che per ogni soluzione ottimale del problema la variabile binaria Y_{jt} assume valore 0 se la corrispondente variabile di produzione P_{jt} risulta nulla.

3.Knapsack

Nel problema di *Knapsack (zaino)* sono assegnati *n* oggetti, per ciascuno dei quali è noto il valore p_j e l'ingombro w_j, j=1, 2, …, *n.* E' inoltre specificata la capacità massima *b* di *knapsack.* Si richiede di determinare un sottoinsieme di oggetti da mettere nello zaino in modo da massimizzare il valore degli oggetti inclusi, rispettando il vincolo di capacità relativo all'ingombro.

Gli ambiti applicativi nei quali si presenta un tale problema sono molteplici: ad esempio, si consideri un corriere che deve caricare un automezzo di capacità *b*, potendo scegliere tra *n* pacchi, ognuno avente un valore p_j e un ingombro w_j, per ottimizzare il valore totale del carico; un secondo esempio è costituito dal problema di *capital budgeting*, in cui si richiede di investire una somma massima di *b* euro in *n* progetti, ciascuno dei quali ha una redditività p_j e comporta un investimento w_j, e bisogna selezionare il sottoinsieme di progetti da attivare per massimizzare la redditività totale, nel rispetto del limite di budget disponibile.

Per risolvere un problema di *knapsack* si costruisce un modello di ottimizzazione lineare intera, introducendo le variabili binarie x_j, j=1,2,...,n,

x_j= 1 se il j-esimo oggetto viene incluso nello zaino
0 altrimenti

e si formula il problema di ottimizzazione binaria pura

$$\max \sum_{j=1}^{n} p_j x_j$$
$$s.a \sum_{j=1}^{n} w_j x_j \leq b$$
$$x_j \in \{0,1\} \qquad j = 1,2,...,n$$

dove le condizioni $x_j \in \{0,1\}$ j=1,2,...,n che descrivono le variabili binarie, possono essere sostituite dalle condizioni equivalenti

$0 \leq x_j \leq 1$, x_j intera j =1,2,...,n.

Una naturale estensione del problema di *knapsack* prevede l'impiego di diversi tipi di risorse disponibili in quantità limitata. Ad esempio, per il problema di *capital budgeting* si suppone che l'attivazione di ciascuno degli n progetti di investimento possa richiedere, accanto al capitale iniziale, l'impiego di risorse umane e tecnologiche, di materie prime e di componenti. Pertanto, al generico progetto j sono associati, oltre al rendimento p_j, anche m coefficienti w_{ij}, i=1,2,...,m che specificano l'assorbimento unitario per ciascuna delle m risorse richieste. Inoltre, si suppone nota la disponibilità totale b_i per ogni risorsa. Si tratta di un problema di *capital budgeting multiplo* in cui si richiede di determinare l'insieme ottimale di progetti da attivare in modo da massimizzare il rendimento totale nel rispetto dei vincoli di disponibilità delle risorse.

Per formulare il problema si utilizza un modello di ottimizzazione binaria pura

$$\max \sum_{j=1}^{n} p_j x_j$$
$$s.a \sum_{j=1}^{n} w_j x_j \leq b_i \qquad i = 1,2,...,m$$
$$x_j \in \{0,1\} \qquad j = 1,2,...,n$$

4.Localizzazione di impianti

Supponiamo che siano assegnati m origini candidate ad ospitare unità produttive, e n destinazioni che devono essere rifornite di prodotti trasportati dalle origini. Ogni origine è caratterizzata da una disponibilità massima di prodotto, a_i, i = 1,2,...,m e da un costo fisso di investimento per attivare un'unità produttiva f_i mentre ad ogni deposito è associata una domanda d_j, j = 1,2,...,n. Il costo unitario di trasporto per inviare un'unità di prodotto dall'origine i al deposito j, per ogni coppia (i,j) di origine e destinazione, è indicato con il simbolo c_{ij}. L'obiettivo consiste nella minimizzazione del costo totale, rappresentato dalla somma dei costi fissi di investimento e dei costi di trasporto, nel rispetto delle capacità massime di deposito delle origini e delle richieste delle destinazioni.
Le variabili di decisione del problema rappresentano le quantità da trasportare da ciascuna origine verso ciascuna destinazione

x_{ij} = quantità di prodotto da trasportare dall'origine i alla destinazione j

cui si aggiungono le variabili binarie

y_i = 1 se l'origine viene attivata

0 altrimenti

Il problema di localizzazione di impianti può essere formulato mediante il seguente modello di ottimizzazione intera mista

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

$$s.a \sum_{j=1}^{n} x_{ij} \leq a_i yi \qquad i = 1,2,...,m$$

$$\sum_{i=1}^{m} xij \geq dj \qquad j = 1,2,...,n$$

$$x_{ij} \geq 0, y_i \in \{0,1\} \qquad i = 1,2,...,m, j = 1,2,...,n$$

5.Scheduling

Consideriamo un sistema produttivo costituito da m macchine che devono essere utilizzate per la produzione di n ordini di lavorazione distinti. Per ogni ordine j = 1,2,...,n è assegnato un limite superiore alla data di consegna d_j, e una durata di lavorazione s_{ji} sulla macchina i, i = 1,2,...,m.
Supponiamo di avere un sistema produttivo denominato *flow-shop*, cioè che la sequenza di

lavorazione sulle diverse macchine sia la stessa per tutti gli ordini. Possiamo numerare le macchine in modo che la *i*-esima lavorazione abbia luogo sulla macchina *i*, *i* = 1,2,...,*m*.
Il problema di *flow-shop scheduling* consiste nel determinare la sequenza di lavorazione ottimale in modo da minimizzare il tempo totale di completamento di tutti gli ordini, nel rispetto delle date di consegna di ognuno di essi.
Si introducono le variabili di decisione

t_{ji} = istante di inizio della lavorazione dell'ordine *j* sulla macchina *i*
T = istante di completamento di tutte le lavorazioni
y_{jki} = 1 se l'ordine *j* viene svolto prima di *k* sulla macchina *i*, con *k* > *i*
0 altrimenti

Indichiamo con Γ una limitazione superiore alla data di completamento di tutte la lavorazioni. Ad esempio, si può porre

$$\Gamma = \sum_{j=1}^{n} \sum_{i=1}^{m} s_{ji}.$$

Il modello da utilizzare è quello di ottimizzazione intera mista

$$\begin{array}{lll} \min T & & \\ s.a \quad t_{jm} + s_{jm} \le T & \forall j & (a) \\ t_{jm} + s_{jm} \le d_j & \forall j & (b) \\ t_{ji} + s_{ji} \le t_{ki} + \Gamma(1 - y_{jki}) & \forall j, \forall k, \forall i & (c) \\ t_{ki} + s_{ki} \le t_{ji} + \Gamma y_{jki} & \forall j, \forall k, \forall i & (d) \\ t_{ji} + s_{ji} \le t_{j,i+1} & \forall j, \forall i \ne m & (e) \\ t_{ji} \ge 0, T \ge 0, y_{jki} \in \{0,1\} & \forall j, \forall k, \forall i & \end{array}$$

I vincoli (*a*) definiscono l'istante di completamento di tutti gli ordini, imponendo che esso sia maggiore dell'istante di completamento dell'ultima lavorazione *m* per ciascun ordine *j*. I vincoli (*b*) determinano il rispetto delle date di consegna, imponendo che per ciascun ordine *j* la lavorazione sull'ultima macchina *m* sia conclusa entro la data prevista d_j .
I vincoli (*c*) e (*d*) hanno lo scopo di impedire la lavorazione simultanea di due ordini *j* e *k* sulla stessa macchina *i*. Se la lavorazione di *j* precede quella di *k* sulla macchina *i*, si attiva il primo dei due vincoli, imponendo che la lavorazione di *j* sia terminata prima che quella di *k* abbia inizio, mentre il secondo vincolo risulta inattivo. Se invece la lavorazione di *k* precede quella di *j* sulla macchina *i*, si considera il vincolo (*d*) imponendo che la lavorazione di *k* sia

terminata prima dell'inizio di quella di *j*, ed è il vincolo (*c*) a risultare inattivo.
I vincoli (*e*), infine, assicurano che per ciascun ordine *j* le lavorazioni procedano nella sequenza prevista sulle *m* macchine.

5.1.Vincoli either-or e disgiuntivi

Il modello del problema di *scheduling* illustra uno schema generale che permette di ricondurre una coppia di vincoli legati da una relazione logica di tipo *either-or*, ovvero uno ed uno solo dei vincoli dev'essere soddisfatto, a una coppia di vincoli in forma congiuntiva legati da una relazione di tipo 'AND'.
La condizione che previene dall'effettuare la lavorazione simultanea degli ordini *j* e *k* sulla stessa macchina può essere espressa ponendo in alternativa tra loro i seguenti vincoli

$$t_{ji} + s_{ji} \leq t_k$$
$$t_{ki} + s_{ki} \leq t_{ji}$$

Per ridurre due vincoli *either-or* alla forma congiuntiva, supponiamo di avere un problema di ottimizzazione nelle variabili $\mathbf{x} \geq \mathbf{0}$, per il quale esistono due vincoli either-or $\mathbf{u'x} \leq b$ e $\mathbf{v'x} \leq d$, con $\mathbf{u} \geq \mathbf{0}$ e $\mathbf{v} \geq \mathbf{0}$, dei quali uno e uno solo deve essere soddisfatto. Definiamo una variabile binaria *y* e sostituiamo la coppia di vincoli in altri in forma congiuntiva, con $\Gamma > 0$ sufficientemente grande,

$$\mathbf{u'x} \leq b + \Gamma y$$
$$\mathbf{v'x} \leq d + \Gamma(1-y)$$
$$y \in \{0,1\}$$

In altri modelli è invece presente una coppia di vincoli disgiuntivi, legati da una relazione logica di tipo 'OR', ovvero almeno un vincolo deve essere soddisfatto. Si preferisce ricorrere a variabili binarie, che consentono di ricondurre una coppia di vincoli disgiuntivi a una di vincoli congiuntivi.
Supponiamo di avere un problema di ottimizzazione nella variabili $\mathbf{x} \geq \mathbf{0}$, per il quale esistono due vincoli disgiuntivi $\mathbf{u'x} \geq b$ e $\mathbf{v'x} \geq d$, con $\mathbf{u} \geq \mathbf{0}$ e $\mathbf{v} \geq \mathbf{0}$, dei quali almeno uno dev'essere soddisfatto. Definiamo una variabile binaria *y* e sostituiamo la coppia di vincoli in altri in forma congiuntiva,

$$\mathbf{u'x} \geq by$$
$$\mathbf{v'x} \leq d(1-y)$$
$$y \in \{0,1\}$$

5.2.Covering, packing e partitioning

Dato un insieme $R = \{r_1, r_2,...,r_m\}$ formato da m elementi e una collezione $S = \{S_1, S_2,...,S_n\}$ di n suoi sottoinsiemi, una sottocollezione di S definita dall'insieme di indici

$$Q \subseteq \{1,2,...,n\}$$

viene detta *covering* se

$$\bigcup_{j \in Q} S_j = R.$$

Se inoltre i sottoinsiemi che formano il *covering* Q sono disgiunti a coppie, ovvero

$$S_j \cap S_k = \emptyset \qquad j,k \in Q, j \neq k$$

la sottocollezione individuata da Q viene detta *partitioning* di R.

Infine, una sottocollezione associata agli indici Q viene detta *packing* se i sottoinsiemi che la compongono sono disgiunti a coppie, ma la loro unione non contiene l'intero insieme R.

$$S_j \cap S_k = \emptyset \qquad j,k \in Q, j \neq k \qquad \bigcup_{j \in Q} S_j \neq R$$

Se per ogni insieme S_j della collezione S è assegnato un peso c_j, il problema di *covering di peso minimo*, o *set-covering*, consiste nel determinare un *covering* per il quale sia minima la somma dei pesi; il problema di *packing di peso massimo*, o *set-packing*, richiede di identificare un *packing* per il quale sia massima la somma dei pesi. Infine, per il problema di *partitioning*, o *set-partitioning*, è possibile considerare sia la massimizzazione che la minimizzazione dei pesi.

I problemi di *set-covering*, *set-packing* e *set-partitioning* si presentano in numerose applicazioni e possono essere formulati come modelli di ottimizzazione intera. Per questo scopo è necessario introdurre una matrice di incidenza **A** = [a_{ij}] costituita da m righe e n colonne come

a_{ij} = 1 se $r_i \in S_j$

0 altrimenti.

Definiamo anche le variabili di decisione binarie

x_j = 1 se $j \in Q$

0 altrimenti.

Il problema di *set-covering di peso minimo* si può formulare come modello di ottimizzazione binaria pura

min **c'x**

s.a **Ax** ≥ **1**

x Є $\{0,1\}^n$.

Come esempio applicativo si può considerare di dover localizzare sul territorio alcuni servizi d'emergenza, quali ambulanze o vigili del fuoco. Suddividiamo la zona che dev'essere servita in *m* aree $R = \{r_1, r_2,...,r_m\}$ e identifichiamo *n* potenziali collocazioni in cui dislocare i servizi. Ciascuna collocazione *j* è in grado di servire un insieme S_j di aree e comporta un peso c_j pari al costo di installazione del servizio *j*. Si vuole identificare una sottocollezione delle localizzazioni che costituisca un *covering* delle aree e che abbia un peso totale minimo.

Il problema di *set-packing di peso massimo* viene formulato come

max **c'x**

s.a **Ax** ≤ **1**

x Є $\{0,1\}^n$.

Infine, si può formulare il problema di *set-partitioning di perso minimo* come

min **c'x**

s.a **Ax = 1**

x Є $\{0,1\}^n$.

Un esempio pratico è dato dal caso di una compagnia aerea che deve fornire i turni di volo degli equipaggi. Sono previste *m* tappe di volo $R = \{r_1, r_2,...,r_m\}$ da effettuarsi in orari prestabiliti. Ciascuna di esse è costituita da una singola fase di volo compresa tra un decollo e un atterraggio. Si procede a concatenare gruppi di tappe di volo che posso costituire un turno ammissibile, tenendo conto di parametri quali la durata complessiva e i periodi di riposo. Ciascun turno S_j è quindi costituito da un sottoinsieme delle tappe e comporta un peso c_j pari al costo del turno. Indichiamo come $S = \{S_1, S_2,...,S_n\}$ la collezione di tutti i potenziali turni. Il problema viene ricondotto a un *set-partitioning di peso minimo*.

6.Assegnazione

Supponiamo di voler assegnare m compiti di lavoro a *n* persone, con *n* = *m*: ciascun compito dev'essere assegnato ad una sola persona e a ciascuna persona dev'essere assegnato un solo compito. Lo svolgimento del compito *i* da parte della persona *j* comporta un costo c_{ij}.

Introduciamo le variabili binarie

x_j = 1 se il compito *i* viene assegnato alla persona *j*

0 altrimenti.

Il problema di assegnazione viene descritto dal modello di ottimizzazione binaria pura

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

$$s.a \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, 2, ..., m$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1, 2, ..., n$$

$$x_{ij} \in \{0, 1\} \qquad i = 1, 2, ..., m,\ j = 1, 2, ..., n.$$

I primi vincoli impongono che ogni compito sia assegnato a una e una sola persona, mentre i secondi che a ogni persona venga affidato uno ed un solo compito.

Se n ≥ m, ossia il numero di persone può essere maggiore di quello dei compiti, è necessario sostituire nei secondi vincoli la relazione di '=' con la relazione di '≤'.

CAPITOLO 2
Proprietà dell'ottimizzazione intera

Consideriamo un generico problema di ottimizzazione lineare intera mista posto in forma di massimizzazione

$$\max \mathbf{c'x + d'y}$$
$$\text{s.a } \mathbf{Ax + Ey = b}$$
$$\mathbf{x} \in \mathbf{Z}^{n}_{+}, \mathbf{y} \geq \mathbf{0}.$$

Per semplificare la notazione, si assumono assenti la varibili continue **y** e quindi abbiamo un problema di ottimizzazione intera pura

$$\max \mathbf{c'x}$$
$$\text{s.a } \mathbf{Ax = b}$$
$$\mathbf{x} \in \mathbf{Z}^{n}_{+}.$$

La verifica di ottimalità e l'arresto degli algoritmi risolutivi per l'ottimizzazione intera vengono ricondotti alla ricerca di una limitazione inferiore $z_L \leq z_I^*$ e di una limitazione superiore $z_U \geq z_I^*$ del valore ottimo z_I^*.

Nella maggior parte dei casi, la limitazione inferiore corrisponde al valore della funzione obiettivo associato alla miglior soluzione determinata dall'algoritmo nel corso delle iterazioni precedenti, mentre per ricavare quella superiore si ricorre al metodo del rilasciamento, che descriveremo più avanti.

Allorché si verifica la condizione di uguaglianza tra le due limitazioni $z_L = z_U$ è possibile asserire l'ottimalità della soluzione ammissibile corrispondente alla limitazione z_L e la validità della relazione $z_I^* = z_L$

La maggior parte degli algoritmi risolutivi ricava una successione di limitazioni inferiori $\{z_{L_k}\}$, corrispondenti alla migliore soluzione ammissibile determinata fino all'interazione k, e una successione di limitazioni superiori $\{z_{u_k}\}$, arrestandosi quando si verifica la condizione $z_{L_k} = z_{u_k}$. Tale condizione non è sempre verificata, così nella maggioranza dei casi applicativi vengono utilizzati degli algoritmi approssimanti, ottenuti mediante una condizione di arresto sul massimo errore assoluto $z_{u_k} - z_{L_k} \leq \varepsilon$ per un opportuno scalare $\varepsilon > 0$ che regola il livello di approssimazione desiderato.

1.Rilasciamenti

Se nel problema

$$\begin{aligned} &\max \mathbf{c'x} \\ &\text{s.a } \mathbf{Ax = b} \\ &\quad \mathbf{x} \in \mathbf{Z}^n_+ . \end{aligned}$$

Vengono rimosse le condizioni di interezza $\mathbf{x} \in \mathbf{Z}^n_+$ relative a tutte le variabili di decisione intere, il problema di ottimizzazione lineare

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

viene detto rilasciamento lineare, o rilasciamento continuo, del problema.

Dato un problema di ottimizzazione lineare intera e il suo rilasciamento continuo, espressi sotto forma di massimizzazione, vale sempre la relazione $z_I^* \leq z_c^*$. Se i due problemi sono in forma di minimizzazione vale invece la relazione $z_I^* \geq z_c^*$.

Di conseguenza, il valore ottimo z_c^* del rilasciamento lineare nel primo caso rappresenta una limitazione superiore per il valore ottimo z_I^* del corrispondente problema a variabili intere. Nel secondo caso invece costituisce una limitazione inferiore per z_I^*.

Si può quindi affermare che se nella soluzione ottimale $\mathbf{x}_C^*$ del rilasciamento continuo tutte le variabili $\mathbf{x}$ assumono valori interi, allora $\mathbf{x}_C^*$ è ottimale anche per il corrispondente problema a variabili intere.

2.Geometria dell'ottimizzazione intera

Consideriamo il seguente problema di ottimizzazione lineare intera

$$\begin{aligned} &\max \quad 8x_1 + 5x_2 \\ &s.a \quad 9x_1 + 5x_2 \leq 45 \\ &\qquad x_1 + 3x_2 \leq 16 \\ &\qquad x_1, x_2 \in \mathbb{Z}_+ . \end{aligned}$$

Per risolverlo trascuriamo la condizione di interezza e riconducendosi ad un problema standard otteniamo

$$\begin{aligned} \min \quad & -8x_1 - 5x_2 \\ & 9x_1 + 5x_2 + x_3 = 45 \\ & x_1 + 3x_2 + x_4 = 16 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Utlizzando la tecnica del simplesso, si scrive il tableau, completo di riga dei costi:

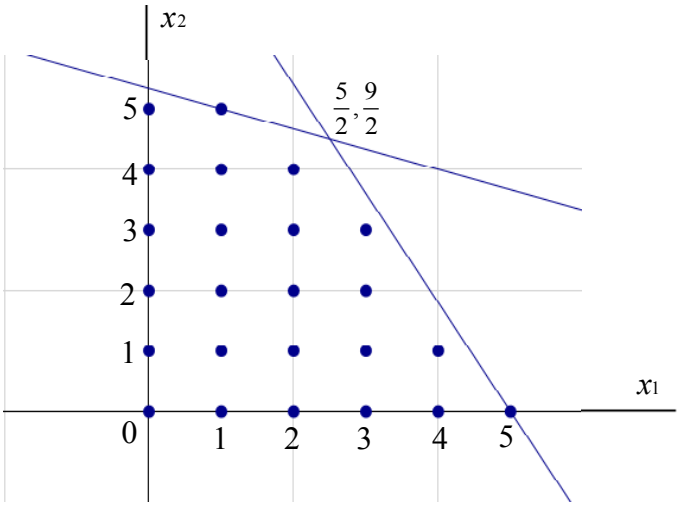
0	-8	-5	0	0
45	9	5	1	0
16	1	3	0	1

Si fa entrare in base x_1 per il criterio del gradiente più negativo e si fa il pivot attorno all'elemento della prima riga perché 45/9 < 16/1.

40	0	$-5/9$	$8/9$	0
5	1	$5/9$	$1/9$	0
11	0	$22/9$	$-1/9$	1

$85/2$	0	0	$19/22$	$5/22$
$5/2$	1	0	$3/22$	$-5/22$
$9/2$	0	1	-1/22	$9/22$

Si ottiene $x_1 = 5/2$, $x_2 = 9/2$ e $z(\mathbf{x}) = 85/2$.



Graficamente la regione ammissibile *P* del rilasciamento lineare è rappresentata dal poliedro convesso in figura.

Il valore ottimo del rilasciamento si trova in corrispondenza del vertice di incontro tra le due rette e risulta $z_c^* = 85/2$.

La regione ammissibile *S* del problema originario è costituita dai soli punti a coordinate intere contenuti nel poliedro, che sono stati evidenziati. La soluzione ottimale del problema di ottimizzazione intera ha coordinate (5,0) cui corrisponde una valore ottimo $z_I^* = 40$.

Osserviamo che in questo caso la soluzione ottimale intera non coincide con la soluzione ottimale del rilasciamento lineare: $z_I^* \neq z_c^*$. Inoltre, z_I^* non coincide con nessuno degli arrotondamenti di z_c^*, rappresentati dai quattro punti di coordinate intere (2,4), (2,5), (3,4), (3,5). Di questi ultimi, il solo punto (2,4) risulta ammissibile.

In generale quindi, la soluzione ottimale del rilasciamento continuo non fornisce né la soluzione ottimale del corrispondente problema a variabili intere né una buon approssimazione di quest'ultima o del valore ottimo intero.

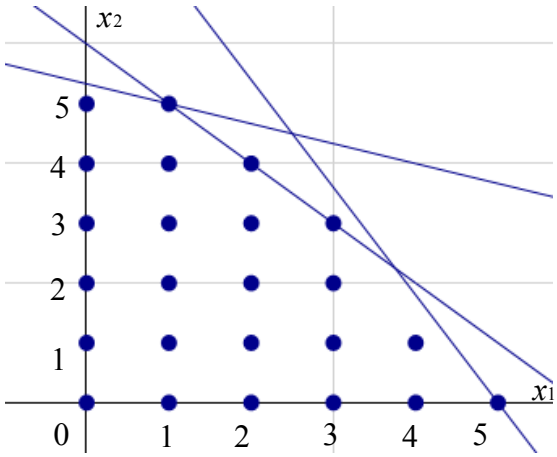
3.Formulazioni alternative e formulazione ideale

Si consideri la regione ammissibile *S* del problema di ottimizzazione intera visto nel paragrafo precedente.

Si può verificare che la medesima regione ammissibile è determinata dai vincoli del seguente problema

$$\begin{aligned} \max \quad & 8x_1 + 5x_2 \\ s.a \quad & 9x_1 + 5x_2 \leq 45 \\ & x_1 + 3x_2 \leq 16 \\ & x_1 + x_2 \leq 6 \\ & x_1, x_2 \in \mathbb{Z}_+ . \end{aligned}$$

che identificano il poliedro illustrato in figura



Pertanto, i due problemi sono equivalenti e il secondo rappresenta una seconda possibile formulazione lineare del primo.

Per definizione, un poliedro $P = \{\mathbf{x} \in R^n: \mathbf{Ax} \leq \mathbf{b}\}$ costituisce una formulazione lineare di un problema di ottimizzazione intera avente regione ammissibile S se vale la relazione $S = P \cap Z^n$.

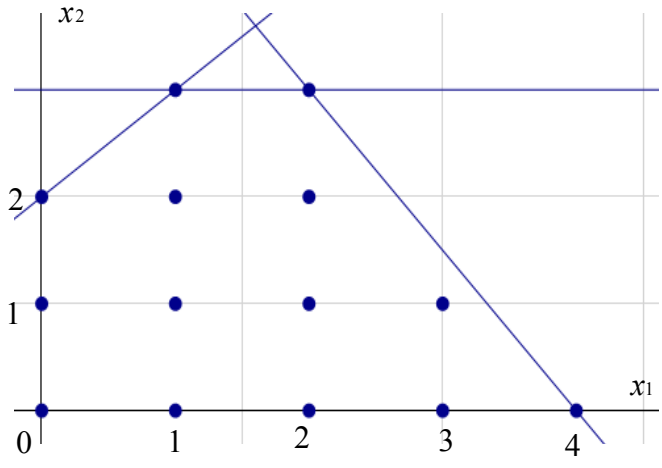
In generale, la regione ammissibile di un problema di ottimizzazione lineare intera può essere ottenuta mediante infinite formulazione alternative equivalenti, al variare degli iperpiani che la delimitano. Date due di queste, caratterizzate rispettivamente dal poliedri P_1 e P_2, diremo che la prima è più stringente della seconda, e quindi migliore, se P_1 appartiene a P_2.

Tra tutte le formulazioni alternative, è possibile identificare come ideale quella per la quale il poliedro convesso associato al rilasciamento lineare possiede tutti i vertici a coordinate intere.

La formulazione ideale del problema analizzato è

$$\begin{array}{ll} \max & x_1 + 2x_2 \\ s.a & -x_1 + x_2 \leq 2 \\ & x_2 \leq 3 \\ & 3x_1 + 2x_2 \leq 12 \\ & x_1, x_2 \in \mathbb{Z}_+ \end{array}$$

e la figura illustra il poliedro associato al suo rilasciamento lineare.



CAPITOLO 3
Metodi risolutivi

1.Metodo dei piani di taglio

Il metodo dei piani di taglio è un algoritmo esatto iterativo per la risoluzione di problemi di ottimizzazione lineare intera pura o mista.
Per semplicità consideriamo un problema di ottimizzazione intera pura

$$z_I^* = \max \mathbf{c'x}$$
$$\text{s.a } \mathbf{Ax = b}$$
$$\mathbf{x} \in \mathbf{Z}^n_+ .$$

E il suo rilasciamento lineare

$$\mathbf{z}_C^* = \max \mathbf{c'x}$$
$$\text{s.a } \mathbf{Ax = b}$$
$$\mathbf{x} \geq \mathbf{0}$$

Questo algoritmo determina dapprima una soluzione ottimale $\mathbf{x}_C^*$ del rilasciamento lineare *C*. Se è intera il metodo si arresta, poiché in tal caso è anche ottimale per *I*. Se invece non fosse intera, viene aggiunto al problema un nuovo vincolo lineare che sia violato da $\mathbf{x}_C^*$ ma che sia soddisfatto da tutte le soluzioni ammissibili di *I*, ottenendo un nuovo problema esteso. In ciascuna iterazione il metodo procede risolvendo il rilasciamento lineare del nuovo problema, e aggiungendo un nuovo vincolo, fino a quando viene raggiunta una soluzione intera.
Il metodo dei piani di taglio rappresenta una famiglia di algoritmi, che differiscono tra loro per la logica utilizzata nella fase di generazione del nuovo vincolo da aggiungere al problema ad ogni iterazione.
Indichiamo con I_k il problema di ottimizzazione lineare intera in corrispondenza della *k*-esima iterazione, e come C_k il corrispondente rilasciamento continuo. I_k viene ottenuto a partire dal problema iniziale $I = I_0$ mediante l'aggiunta di *k* nuovi vincoli, e quindi si presenta nella forma

$$z_{Ik}^* = \max \mathbf{c'x}$$
$$\text{s.a } \mathbf{Ax = b}$$
$$\boldsymbol{v'_h x} = g_h \qquad h = 1,2,...,k$$
$$\mathbf{x} \in \mathbf{Z}^n_+ ,$$

dove $\boldsymbol{v'_h x} = g_h$, h = 1,2,...,k è l'insieme dei vincoli introdotti nel corso delle iterazioni precedenti. Abbiamo implicitamente assunto di aggiungere un nuovo vincolo per ciascuna iterazione, ma gli argomenti sviluppati rimangono validi anche se vengono introdotti più vincoli simultaneamente.

Supponiamo ora che la soluzione ottimale $\mathbf{x}_{Ck}^*$ di C_k non sia intera. Il vincolo $\boldsymbol{v'_{k+1} x} = g_{k+1}$ viene detto taglio valido per il problema I_k se valgono le seguente proprietà:

- il vincolo è violato dalla soluzione ottimale $\mathbf{x}_{Ck}^*$ di C_k ;
- il vincolo è soddisfatto da tutte le soluzioni ammissibili di I.

Da un punto di vista geometrico, il vincolo costituisce un taglio valido se elimina dalla regione ammissibile del nuovo rilasciamento lineare la soluzione ottimale del precedente rilasciamento, senza escludere nessun punto intero del problema originario. Mediante la generazione di tagli validi l'algoritmo preserva la regione ammissibile S del problema I anche per tutti i problemi I_k . In altri termini, viene generata una successione di formulazioni alternative equivalenti del problema I.

1.1.Finitezza dell'algoritmo e interruzione prematura.

Osserviamo $\boldsymbol{v'_{k+1} x} = g_{k+1}$ che la procedura può proseguire all'infinito se non si verifica la condizione di arresto, cioè se non viene raggiunta una soluzione ottimale intera. Esistono tuttavia regole di generazione dei tagli validi per le quali è possibile dimostrate la finitezza dell'algoritmo. Tra queste, il più importante è il metodo dei tagli di Gomory.

Anche tali metodi però comportano un aumento considerevole del numero di tagli aggiuntivi al progredire delle iterazioni, e un conseguente incremento esponenziale dei tempi di calcolo.

Il metodo dei piani di taglio, inoltre, presenta un difetto: poiché la soluzione ottimale $\mathbf{x}_{Ck}^*$ di ciascuno dei rilasciamenti lineari C_k è frazionaria, e quindi non ammissibile per I in tutte le iterazioni k ad eccezione dell'ultima, l'interruzione prematura dell'algoritmo non fornisce una soluzione ammissibile intera utilizzabile come approssimazione della soluzione ottimale $\boldsymbol{x}_I^*$.

1.2.Taglio di Gomory.

Descriviamo la regola di Gomory per la generazione di un taglio valido $\boldsymbol{v'_{k+1} x} = g_{k+1}$ illustrando parallelamente un esempio applicativo.

$$\begin{aligned} \max\ & x_2 \\ & 3x_1 + 2x_2 \le 6 \\ & -3x_1 + 2x_2 \le 0 \\ & x_1, x_2 \in \mathbb{Z}_+^n \end{aligned}$$

Si procede risolvendo il problema rilassato (si trascura il fatto che le variabili devono essere intere), comportandosi quindi come se si trattasse di un normale problema di programmazione lineare continua. Trasformo il problema in forma standard

$$\begin{aligned} \min \quad & -x_2 \\ & 3x_1 + 2x_2 + x_3 = 6 \\ & -3x_1 + 2x_2 + x_4 = 0 \\ & x_1, \ldots, x_4 \geq 0 \end{aligned}$$

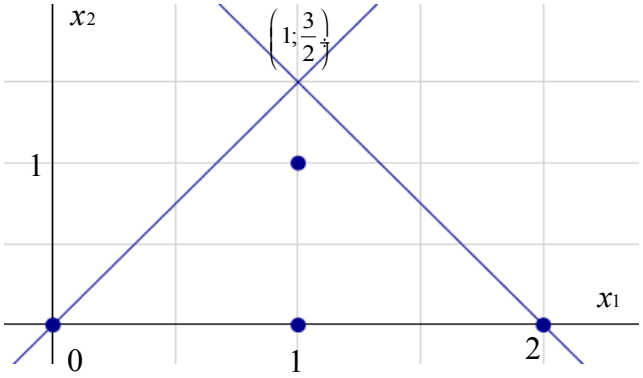
Risolviamo il problema rilassato con il metodo del simplesso:

0	0	-1	0	0
6	3	2	1	1
0	-3	2	0	1

0	0	-1	0	0
2	1	2 / 3	1 / 3	0
6	0	4	1	1

3 / 2	0	0	1 / 4	1 / 4
1	1	0	1 / 6	- 1 / 6
3 / 2	0	1	1 / 4	1 / 4

Quella appena ottenuta, è la soluzione ottima del rilasciamento continuo $\mathbf{x}_{C1}^*$ ($x_1 = 1$; $x_2 = 3/2$), ma non va bene nel nostro caso perché non è intera; graficamente si ha



Considerando l'ultimo tableau, i vincoli che abbiamo sono:

- $x_1 + 1/6\ x_3 - 1/6\ x_4 = 1$
- $x_2 + 1/4\ x_3 + 1/4\ x_4 = 3/2$

Il metodo del taglio di Gomory parte da un vincolo che ha termine noto non intero: prendiamo quindi il secondo vincolo e scomponiamo i coefficienti in parte intera e parte frazionaria:
$x_2 + 1/4\ x_3 + 1/4\ x_4 = 1 + 1/2$.

Porto a destra la parte intera:
$1/4\ x_3 + 1/4\ x_4 = 1 + 1/2 - x_2$.

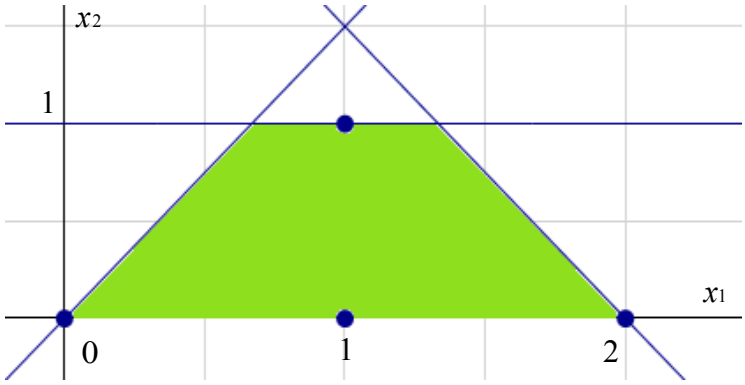
Tralasciando la parte intera si ottiene un taglio di Gomory:
$1/4\ x_3 + 1/4\ x_4 \geq ½$

Ricavo x_3 dal primo vincolo del problema iniziale $3x_1 + 2x_2 + x_3 = 6$:
$x_3 = 6 - 3x_1 - 2x_2$
e x_4 dal secondo vincolo $-3x_1 + 2x_2 + x_4 = 0$:
$x_4 = 3x_1 - 2x_2$.

Li sostituisco nel taglio ottenuto:
$1/4*(6 - 3x_1 - 2x_2) + 1/4*(3x_1 - 2x_2) \geq ½$
ottenendo un nuovo vincolo da aggiungere nel problema originario:
$x_2 \leq 1$.

$$\begin{aligned} \min\ & -x_2 \\ & 3x_1 + 2x_2 + x_3 = 6 \\ & -3x_1 + 2x_2 + x_4 = 0 \\ & x_2 + x_5 = 1 \\ & x_1, \ldots, x_5 \geq 0 \end{aligned}$$

Graficamente otteniamo:



Come si può osservare, si può verificare che il nuovo vincolo corrisponde ad un taglio valido in quanto:

- il vincolo è violato dalla soluzione ottimale del precedente rilasciamento lineare (una parte della regione ammissibile del problema precedente è stata esclusa);
- il vincolo è soddisfatto da tutte le soluzioni ammissibili del problema originario (i punti a coordinate intere del problema sono incluse nello spazio delle soluzioni del problema con l'aggiunta del taglio).

Si proseguono ora i calcoli con il nuovo vincolo:

3 / 2	0	0	1 / 4	1 / 4	0
1	1	0	1 / 6	- 1 / 6	0
3 / 2	0	1	1 / 4	1 / 4	0
1	0	1	0	0	1

Sottraggo alla quarta riga la terza riga

3 / 2	0	0	1 / 4	1 / 4	0
1	1	0	1 / 6	- 1 / 6	0
3 / 2	0	1	1 / 4	1 / 4	0
- 1 / 2	0	0	- 1 / 4	- 1 / 4	1

1	0	0	0	0	1
2 / 3	1	0	0	- 1 / 3	2 / 3
1	0	1	0	0	1
2	0	0	1	1	-4

La soluzione ottima del rilasciamento continuo $\mathbf{x}_{C2}^*$ ($x_1 = 2/3$; $x_2 = 1$; $x_3 = 2$) non è intera; è necessario quindi continuare la procedura di generazione di tagli validi.

Abbiamo questi vincoli:

- $x_1 - 1/3\ x_4 + 2/3\ x_5 = 2/3$
- $x_2 + 1/2\ x_4 + x_5 = 1$
- $x_3 + x_4 - 4x_5 = 2$

Considero il primo vincolo perché ha termine noto non intero e lo scompongo in parte intera e frazionaria:
$x_1 - x_4 + 2/3\ x_4 + 2/3\ x_5 = 2/3$
$2/3\ x_4 + 2/3\ x_5 = 2/3 + x_4 - x_1$

Tralascio la parte intera e risulta:
$2/3\ x_4 + 2/3\ x_5 \geq 2/3$

Avevamo già visto che
$x_3 = 6 - 3x_1 - 2x_2$
e $x_4 = 3x_1 - 2x_2$.

Inoltre, dal primo taglio $1/4\ x_3 + 1/4\ x_4 - x_5 = 1/2$
ottengo che $x_5 = -1/2 + 1/4\ x_3 + 1/4\ x_4$.

Sostituisco le espressioni in $2/3\ x_4 + 2/3\ x_5 \geq 2/3$:
$2x_1 - 2x_2 \geq 0$.
Questo è il nuovo vincolo da aggiungere ai precedenti.

$$\begin{aligned} &\min -x_2 \\ &3x_1 + 2x_2 + x_3 = 6 \\ &-3x_1 + 2x_2 + x_4 = 0 \\ &x_2 + x_5 = 1 \\ &2x_1 - 2x_2 - x_6 = 0 \\ &x_1, \ldots, x_6 \geq 0 \end{aligned}$$

Per semplicità di calcoli aggiungiamo in tabella il vincolo corrispondente al taglio di Gomory, senza aver sostituito le variabili x_4 e x_5:
$2/3\ x_4 + 2/3\ x_5 - x_6 = 2/3$

1	0	0	0	0	1	0
2/3	1	0	0	-1/3	2/3	0
1	0	1	0	0	1	0
2	0	0	1	1	-4	0
2/3	0	0	0	2/3	2/3	-1

Cambiamo di segno l'ultimo vincolo:

1	0	0	0	0	1	0
2/3	1	0	0	-1/3	2/3	0
1	0	1	0	0	1	0
2	0	0	1	1	-4	0
-2/3	0	0	0	-2/3	-2/3	1

1	0	0	0	0	1	0
1	1	0	0	0	1	-1/2
1	0	1	0	0	1	0
1	0	0	1	0	-5	3/2
1	0	0	0	1	1	-3/2

La soluzione ottima del rilasciamento continuo $\mathbf{x}_{C3}^*$ ($x_1 = 1$; $x_2 = 1$; $x_3 = 1$; $x_4 = 1$) è intera e corrisponde quindi alla soluzione ottima del problema originario di ottimizzazione lineare intera $\boldsymbol{x}_I^*$ = (1;1) cui è associato un valore della FO pari a 1.

2.Branch and Bound

Un algoritmo alternativo per la risoluzione di un problema di programmazione lineare intera è noto come tecnica di enumerazione implicita, o *branch and bound*. Si tratta di un procedimento del tipo *divide and conquer* che riconduce la risoluzione di un problema 'difficile' alla risoluzione di due sottoproblemi più semplici.

Sia

$$\begin{aligned} &\min \ \mathbf{c'x} \\ &\text{s.a } \mathbf{Ax = b} \\ &\quad \mathbf{x} \in \mathbf{Z}^n_+ \end{aligned}$$

un problema di programmazione intera. Si risolva il rilasciamento continuo

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

individuando una soluzione base ottima $\mathbf{x}^*$.

Se $\mathbf{x}^*$ è intera, non è necessario procedere ulteriormente. Altrimenti, si sceglie una variabile x_h^* frazionaria e si costuiscono i due sottoproblemi:

$$\begin{aligned} &PLI_2\text{: } \min \ \mathbf{c'x} \\ &\quad \text{s.a } \mathbf{Ax = b} \\ &\qquad x_h \leq \lfloor x_h^* \rfloor \\ &\qquad \mathbf{x} \in \mathbf{Z}^n_+ \end{aligned}$$

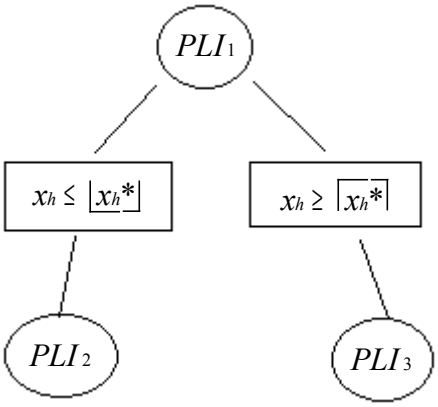
dove il simbolo $\llcorner\!\lrcorner$ corrisponde al primo intero minore di x_h^*,

e

$$\begin{aligned} &PLI_3\text{:} \min \ \mathbf{c'x} \\ &\quad \text{s.a } \mathbf{Ax = b} \\ &\qquad x_h \geq \lceil x_h^* \rceil \\ &\qquad \mathbf{x} \in \mathbf{Z}^n_+ \end{aligned}$$

dove il simbolo $\sqcap\!\!\sqcap$ indica il primo intero maggiore di x_h^*.

Questa operazione viene detta di *branching* (ramificazione) e può essere rappresentata in modo grafico come in figura.



La variabile x_h viene detta variabile di *branching.* I problemi PLI_2 e PLI_3 si dicono figli di PLI_1 e sono ottenuti da questo aggiungendo un vincolo di *branching*.

A questo punto si risolvono i due problemi figli, in sequenza, riapplicando ricorsivamente il procedimento: per ciascuno dei due sottoproblemi, si risolve il rilassamento lineare e se necessario si effettua ancora l'operazione di *branching*. Si ottiene così una successione di problemi via via più vincolati, quindi più facili da risolvere. Raggiunta l'interezza in ciascuno dei rami, si sceglierà la soluzione di costo minimo.

Poiché, per ipotesi, la regione ammissibile $P = \{\mathbf{x} \geq 0\ \mathbf{Ax = b}\}$ è limitata, il numero totale dei nodi (cioè dei sottoproblemi) dell'albero decisionale è finito. Nel caso peggiore, tuttavia, questo numero può essere elevatissimo.

2.1.Metodi di fathoming

Nella pratica non è necessario considerare esplicitamente tutti i nodi potenzialmente presenti nell'albero decisionale. Infatti, alcuni dei nodi foglia (i nodi a cui corrispondono i sottoproblemi) possono corrispondere a problemi il cui rilasciamento continuo è impossibile. Ciò avviene quando i vincoli iniziali $\mathbf{Ax = b,}\ \mathbf{x} \geq 0$, risultano incompatibili con i vincoli di *branching* relativi al percorso che va dal nodo radice (quello relativo al problema iniziale) al nodo foglia in questione. Questi problemi ovviamente non generano dei figli.

Lo stesso accade quando il rilasciamento continuo *C* del nodo *t* corrente ha soluzione ottima $\mathbf{x}^*_C$ intera. Siano infatti:

- $\boldsymbol{x}_{OPT}$ = soluzione ottima corrente (la migliore soluzione intera fin qui trovata)
- $\boldsymbol{z}_{OPT} = \mathbf{c'x}_{OPT}$ = costo della soluzione ottima corrente.

Se $\mathbf{c'x}^*_C < \mathbf{c'x}_{OPT}$ allora si aggiorna $\boldsymbol{z}_{OPT} = \mathbf{c'x}^*_C$ e $\boldsymbol{x}_{OPT} = \mathbf{x}^*_C$; in ogni caso non è necessario analizzare ulteriormente il nodo *t* corrente, e si considerano altri nodi dell'albero decisionale.

Esiste un terzo criterio, detto di *bounding*, che permette spesso di eliminare un gran numero di nodi dell'albero decisionale. Supponiamo di aver risolto il rilassamento continuo Pl_t del

problema associato al nodo corrente *t*, e sia $\mathbf{x}^*_{Ct}$ la soluzione ottima individuata. Se questa è frazionaria dovremmo procedere alla generazione dei figli di *t*, nella speranza che essi permettano di individuare una soluzione intera migliore della soluzione ottima corrente $\boldsymbol{x}_{OPT}$. Poiché però il valore $\mathbf{c'x}^*_{Ct}$ costituisce una stima ottimistica (*lower bound*) del valore della miglior soluzione intera ottenibile a partire dal nodo *t*, se $\mathbf{c'x}^*_{Ct} \geq \boldsymbol{z}_{OPT}$ questa soluzione non potrà essere migliore di quella corrente, e quindi non è necessario elaborare ulteriormente il nodo.

2.2.Realizzazione dell'algoritmo

Nella realizzazione dell'algoritmo dev'esserci una regola con cui selezionare il nodo da elaborare dell'iterazione corrente. Vi sono due tecniche principali:

- tecnica *depht first* (prima i nodi più profondi): dato un nodo padre si considera immediatamente il primo dei suoi due figli, finché una delle condizioni di *fathoming* non costringe l'algoritmo a risalire di livello;
- tecnica best-bound first (prima i nodi più promettenti): la scelta del nodo da elaborare viene fatta scegliendo quello non ancora elaborato con $\mathbf{c'x}^*_{Ct}$ minimo e quindi verosimilmente più vicino alla soluzione ottima.

2.3.Esempio

Consideriamo il seguente problema di programmazione lineare intera:

$$\begin{aligned} \min\ & x_1 + 2x_2 \\ & x_1 + x_2 \leq 4 \\ & x_1 + 4x_2 \geq 6 \\ & 8x_1 - 8x_2 \geq 3 \\ & x_1, x_2 \in \mathbb{Z}^n_+ \end{aligned}$$

Trasformiamo in forma standard:

$$\begin{aligned} \min\ & x_1 + 2x_2 \\ & x_1 + x_2 + x3 = 4 \\ & x_1 + 4x_2 - x_4 = 6 \\ & 8x_1 - 8x_2 - x_5 = 3 \\ & x_1, \ldots, x_5 \geq 0 \end{aligned}$$

e cambiamo i segni alle righe di vincolo in modo da ottenere una forma canonica

$$\begin{aligned} \min\ & x_1 + 2x_2 \\ & x_1 + x_2 + x3 = 4 \\ & -x_1 - 4x_2 + x_4 = -6 \\ & -8x_1 + 8x_2 + x_5 = -3 \\ & x_1, \ldots, x_5 \geq 0. \end{aligned}$$

Scriviamo il tableau

∞	1	2	0	0	0
4	1	1	1	0	0
-6	-1	-4	0	1	0
-3	-8	8	0	0	1

-3	1/2	0	0	1/2	0
5/2	3/4	0	1	1/4	0
3/2	1/4	1	0	-1/4	0
-15	-10	0	0	2	1

-15/4	0	0	0	3/5	1/20
11/8	0	0	1	2/5	3/40
9/8	0	1	0	-1/5	1/40
3/2	1	0	0	-1/5	-1/10

La soluzione ottima del rilasciamento continuo $\mathbf{x}_{C1}^*$ ($x_1 = 3/2$; $x_2 = 9/8$; $x_3 = 11/8$) non è intera e il valore della funzione obiettivo associata a questa soluzione è $z_{C1} = 15/4$.
Abbiamo tre possibili scelte per la variabile di *branching* x_h. Conviene di solito scegliere quella la cui parte frazionaria è più vicina a ½, in modo che il vincolo aggiunto sia significativo per entrambi i sottoproblemi figli. Scegliamo quindi la variabile x_1 e generiamo i nodi 2 e 3.

<u>Nodo 2:</u>
Partendo dal tableau ottimo del nodo padre aggiungiamo il vincolo $x_1 \leq 1$, cioè $x_1 + x_6 = 1$. Ricaviamo x_1 dalla terza riga $x_1 - 1/5\, x_4 - 1/10\, x_5 = 3/2$ ottenendo $x_1 = 3/2 + 1/5\, x_4 + 1/10\, x_5$ e sostituendolo. Il tableau corrispondente è:

-15/4	0	0	0	3/5	1/20	0
11/8	0	0	1	2/5	3/40	0
9/8	0	1	0	-1/5	1/40	0
3/2	1	0	0	-1/5	-1/10	0
-1/2	0	0	0	1/5	1/10	1

3 / 2	0	0	0	0	- 1 / 4	-3
3 / 8	0	0	1	0	- 1 / 8	-2
5 / 8	0	1	0	0	1 / 8	1
1	1	0	0	0	0	1
- 5 / 2	0	0	0	1	1 / 2	5

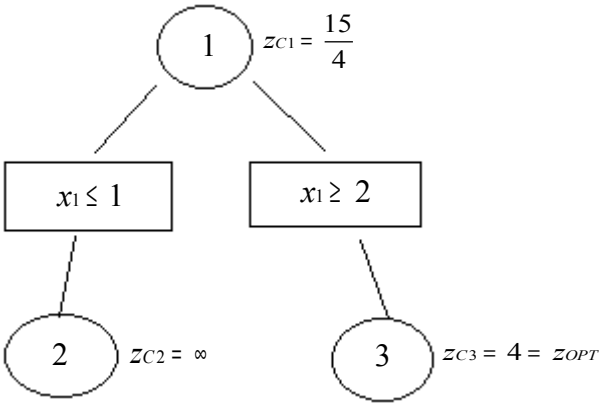
Tale problema risulta impossibile: si passa quindi alla risoluzione dell'altro sottoproblema.

Nodo 3:
Partiamo ancora del tableau ottimo del nodo padre, aggiungendo questa volta il vincolo $x_1 \geq 2$, cioè $x_1 - x_6 = 2$. Ricavando x_1 come prima, sostituendo e infine cambiando segno alla riga si ottiene:

- 15 / 4	0	0	0	3 / 5	1 / 20	0
11 / 8	0	0	1	2 / 5	3 / 40	0
9 / 8	0	1	0	- 1 / 5	1 / 40	0
3 / 2	1	0	0	- 1 / 5	- 1 / 10	0
- 1 / 2	0	0	0	- 1 / 5	- 1 / 10	1

-4	0	0	0	1 / 2	0	1 / 2
1	0	0	1	1 / 4	0	3 / 4
1	0	1	0	- 3 / 20	0	1 / 4
2	1	0	0	0	0	-1
5	0	0	0	2	1	-10

La soluzione ottima del rilasciamento continuo $\mathbf{x}_{C3}^*$ ($x_1 = 2$; $x_2 = 1$; $x_3 = 1$; $x_5 = 5$) è intera e il valore della funzione obiettivo associata a questa soluzione è $z_{C3} = 4$.
Si può quindi aggiornare il valore di $z_{OPT} = 4$. Non è quindi necessario generare i figli del nodo 3: l'algoritmo si interrompe qui.
Rappresentiamo il problema graficamente:



2.4.Enumerazione implicita

Una speciale procedura di Branch and Bound può essere utilizzata per programmazioni intere con solo variabili binarie. L'algoritmo ha il vantaggio di non richiedere soluzioni di programmazione lineare. E' illustrato tramite il seguente esempio:

$$\begin{aligned} \max \quad & -8x_1 - 2x_2 - 4x_3 - 7x_4 - 5x_5 + 10 \\ & -3x_1 - 3x_2 + x_3 + 2x_4 + 3x_5 \leq -2 \\ & -5x_1 - 3x_2 - 2x_3 - x_4 + x_5 \leq -4 \\ & xj = 0 \ \ or \ \ 1 \qquad j = 1,2,...,5. \end{aligned}$$

Un modo per risolvere tale problema è l'enumerazione implicita. E' necessario fare una lista di tutte le possibili combinazioni della variabili e selezionare la migliore combinazione ottenibile. L'approccio è molto utile per un problema semplice come questo, dove ci sono poche potenziali combinazioni 0-1 delle variabili, nel nostro caso 32. In generale, comunque, un problema con *n* variabili contiene 2^n combinazioni 0-1: per alti numeri di *n*, l'approccio esaustivo è quindi proibitivo. Invece, si può considerare implicitamente ogni combinazione binaria, proprio come ogni punto intero è implicitamente considerato, ma non necessariamente valutarlo, per il problema generale risolvibile con il metodo branch and bound.

Ricordiamo che con la procedura ordinaria branch and bound le suddivisioni erano analizzate mantenendo i vincoli lineari e annullando le restrizioni lineari. Qui, adottiamo la tattica opposta di mantenere sempre le restrizioni 0-1, ma ignorando i vincoli di disuguaglianza.

L'idea è quella di utilizzare un processo branch and bound per fissare alcune delle variabili a 0 o 1. Le variabili che rimangono da specificare vengono chiamate variabili libere. Da notare è che, se i vincoli di disuguaglianza vengono ignorati, la funzione obiettivo è massimizzata impostando le variabili libere a 0, quando i coefficienti della funzione obiettivo a loro associati

sono negativi. Per esempio, se x_1 e x_4 sono fissate a 1 e x_5 a 0, le variabili libere sono x_2 e x_3. Ignorando i vincoli di disuguaglianza, il problema risultante è:

$$\max[-8(1)-2x_2-4x_3-7(1)-5(0)+10] = \max[-2x_2-4x_3-5].$$

avente x_2 e x_3 binari.

Finché le variabili x_2 e x_3 hanno coefficienti negativi, la massimizzazione pone $x_2 = 0$ e $x_3 = 0$. La semplicità di questa banale ottimizzazione, comparata ad una programmazione lineare più efficiente, è quella che vogliamo utilizzare.

Ritornando all'esempio, partiamo senza variabili fissate, e conseguentemente ogni variabile è libera e posta a 0. La soluzione non soddisfa i vincoli, quindi dobbiamo ramificare per ricercane una fattibile. Una prima suddivisione può essere:

Ramo 1: $x_1 = 1$,

Ramo 2:

Ora la variabile x_1 è fissata in ogni ramo. Secondo le nostre osservazioni sopra, se le disuguaglianze tra variabili sono ignorate, la soluzione ottimale consiste nel porre uguale a 0 x_2, x_3, x_4, x_5. Il risultato dello svolgimento del primo ramo è dunque

$$z = -8(1)-2(0)-4(0)-7(0)-5(0)+10 = 2.$$

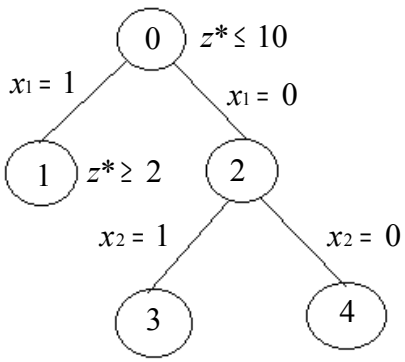
e sembra soddisfare i vincoli, così la soluzione ottimale al problema originale è almeno 2, $z^* \geq 2$. In questo modo il ramo 1 è stato arrestato. La soluzione ottenuta è la migliore tra tutte le combinazioni 0-1 con $x_1 = 1$; questa dev'essere la migliore tra quelle che soddisfano i vincoli Nessun'altra combinazione 0-1 nella suddivisione 1 dev'essere valutata esplicitamente. Queste sono state considerate implicitamente.

La soluzione con $x_2 = x_3 = x_4 = x_5 = 0$ nel secondo ramo è la stessa ottenuta all'origine, e abbiamo visto che non è ammissibile. Conseguentemente, la regione dev'essere subito suddivisa, separando $x_2 = 1$ da $x_2 = 0$, ottenendo:

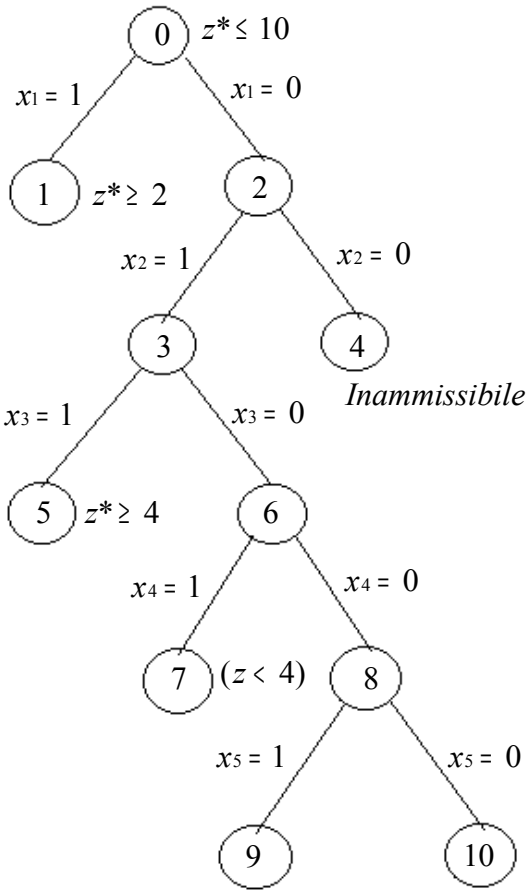
Ramo 3: $x_1 = 0$, $x_2 = 1$,

Ramo 4: $x_1 = 0$, $x_2 = 0$.

L'albero decisionale fino a questo punto è dato dalla figura



Osserviamo che l'albero differisce da quello del branch and bound classico analizzato precedentemente. Per la procedura di prima, si analizza esplicitamente ogni ramo; qui le combinazioni 0-1 che dovrebbero essere analizzate in ogni suddivisione non sono scritte esplicitamente, poiché si sa che le variabili non specificate vengono poste semplicemente pari a 0. Nel ramo 3, per esempio, $x_1 = 0$ e $x_2 = 1$ sono fissate, x_2, x_3 e x_5 sono azzerate.
Continuando a fissare le variabili e suddividendo utilizzando questa procedura l'albero di enumerazione completo è questo:



L'albero non è esteso dopo l'analisi dei nodi 4, 5, 7, 9 e 10, per le seguenti ragioni:

- nel nodo 5, la soluzione $x_1 = 0$, $x_2 = x_3 = 1$, con le variabili $x_4 = x_5 = 0$, è ammissibile, con z = 4: questo è un migliore limite minimo di z*;
- nel nodo 7, la soluzione $x_1 = x_3 = 0$, $x_2 = x_4 = 1$, e la variabile libera $x_5 = 0$, ha z = 1 <4, così nessuna soluzione nei rami che partono da questa può essere migliore della soluzione del nodo 5;
- nei nodi 9 e 10, ogni variabile libera è fissata. In ogni caso, la suddivisione contiene solo un singolo punto, che è inammissibile, e ogni altra ramificazione non è possibile;

– nel nodo 4, la seconda disuguaglianza (con le variabili fissate a $x_1 = x_2 = 0$) è:

$$-2x_3 - x_4 + x_5 \leq -4.$$

Nessun valore 0-1 di x_3, x_4 o x_5 che completano la soluzione insieme alle variabili già fissate, soddisfa questo vincolo. Il nodo non ha quindi soluzioni ammissibili e non deve essere ulteriormente analizzato.

Le tecniche utilizzate qui possono essere applicate a qualsiasi altro problema di programmazione intera avente solo variabili binarie, così che l'enumerazione implicita è una procedura branch and bound alternativa per tale classi di problemi. In questo caso, i rami sono interrotti se qualcuna delle tre condizioni è sviluppata:

– si sa già che la programmazione intera è inammissibile attraverso la suddivisione, per esempio, fatta tramite il test di inammissibilità;

– le soluzioni 0-1 ottenute ponendo le variabili libere a 0 soddisfano i vincoli lineari; o

– il valore obiettivo ottenuto azzerando le variabili libere non è più grande del migliore risultato ammissibile precedentemente generato.

Queste condizioni corrispondono alle tre condizioni di arresto dell'algoritmo nella procedura branch and bound classica. Se una regione non è interrotta da una delle tre condizioni, l'enumerazione implicita suddivide quella regione selezionando una delle variabili libere e fissando il suo valore a 0 o 1.

Le nostre argomentazioni sull'algoritmo trovano fondamento se il problema binario originario si presenta nella seguente forma standard:

1 l'obiettivo è una massimizzazione con tutti coefficienti negativi, e

2 i vincoli sono disuguaglianze del tipo 'minore o uguale di'.

I problemi di minimizzazione sono trasformati in massimizzazioni moltiplicando i coefficienti di costo per -1.

Come nel metodo branch and bound standard per problemi di programmazione lineare intera, il modo in cui scegliamo di suddividere le regioni possono avere profondi effetti sulle combinazioni. Nell'enumerazione implicita, iniziamo con la soluzione zero $x_1 = x_2 = x_3 = x_4 = x_5 = 0$ e generiamo altre soluzioni ponendo le variabili peri a 1. Un approccio naturale è di ramificare basandosi sulle variabili con maggiori contributi nella funzione obiettivo. Nel caso di prima, avremmo dovuto quindi iniziare l'analisi da x_2. Un altro approccio molto utilizzato nella pratica è provare a raggiungere l'ammissibilità prima possibile. Per esempio, quando $x_1 = 0$, $x_2 = 1$, e $x_3 = 0$ sono fissate nell'esempio considerato sopra, possiamo suddividere

basandoci su x_4 e x_5. Ponendo x_4 e x_5 pari a 1 vediamo che i vincoli diventano:

$$x_4 = 1, \quad x_5(libera) = 0$$
$$-3(0) - 3(1) + (0) + 2(1) + 3(0) \leq -2,$$
$$-5(0) - 3(1) - 2(0) - 1(1) + (0) \leq -4,$$

$$x_5 = 1, \quad x_4(libera) = 0$$
$$-3(0) - 3(1) + (0) + 2(0) + 3(1) \leq -2,$$
$$-5(0) - 3(1) - 2(0) - 1(0) + (1) \leq -4.$$

Per $x_4 = 1$, il primo vincolo è inammissibile per un'unità e il secondo ammissibile, dando un punto totale di inammissibilità. Per $x_5 = 1$ il primo vincolo è inammissibile per due unità e il secondo sempre inammissibile per due unità, dando un totale di 4 unità di inammissibilità. Quindi $x_4 = 1$ appare più favorevole, e potremmo ramificare partendo da quella variabile. In generale, la variabile che da la totale più piccola inammissibilità attraverso questo approccio dovrebbe essere scelta per continuare l'albero. Rivedendo l'esempio fatto sopra, si vede come tale metodo è stato utilizzato nel raggiungimento della nostra soluzione.

3.Branch and Cut

Si tratta di una tecnica mista *branch and bound*/metodo dei piani di taglio proposta per superare alcuni degli inconvenienti delle due metodologie.

E' necessario progettare un algoritmo *branch and bound* in cui ad ogno nodo *t* dell'albero decisionale si generano alcuni tagli profondi nella speranza di ottenere una soluzione $\mathbf{x}_C^*$ intera o anche soltanto un lower bound $\mathbf{c'x}^*_{Ct}$ più elevato e quindi più utile per il fathoming. Non appena si osserva che questi tagli diventano poco efficaci, se ne sospende la generazione per effettuare il *branching*.

Rispetto al *branch and bound* puro, si ha il vantaggio di un rafforzamento dinamico della formulazione del problema. Rispetto al metodo dei piani di taglio puro, invece, si ha la possibilità di contrastare mediante l'operazione di *branching* il fenomeno del *taling off* (una lunga serie di iterazioni senza un sostanziale miglioramento della formulazione corrente).

La realizzazione pratica di quest'idea non è immediata. Per esempio, si supponga di voler utilizzare i tagli di Gomory all'interno di uno schema *branch and bound*. I tagli generati sono ricavati dal tableau ottimo associato al nodo dell'albero decisionale corrente, quindi la loro validità dipende anche dai vincoli di *branching* imposti. Ne consegue che, in generale, questi

tagli sono validi solo localmente, cioè per il nodo corrente e per i suo eventuali figli descendenti.

Immaginiamo invece di essere in grado di generare tagli validi globalmente lungo tutto l'albero decisionale. E' allora possibile memorizzare tutti i tagli in un'unica struttura dati generale, chiamata *pool* di vincoli. Elaborando un nuovo nodo, si parte dalla formulazione $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq 0$, si aggiungono i vincoli di *branching* e si risolve il rilassamento continuo corrispondente ottenendo la soluzione $\mathbf{x}^*$. Se questa è frazionaria, si scandisce il pool alla ricerca di vincoli violati da $\mathbf{x}^*$ da aggiungere alla formulazione corrente. Si risolve allora il nuovo problema e si procede così finché $\mathbf{x}^*$ non soddisfa tutti i vincoli del *pool*. Se $\mathbf{x}^*$ è ancora frazionaria, si attiverà una conveniente procedura di separazione per l'individuazione di nuovi tagli globali da inserire nel *pool*.

Il procedimento descritto prende il nome di *branch and cut*. Esso è caratterizzato dalla presenza del *pool* globale, e da una o più procedure di separazione in grado di generare tagli validi globalmente.

La definizione di procedure efficaci di separazione è uno dei punti cruciali del metodo. L'ideale sarebbe disporre di procedure di tipo generale, che siano cioè applicabili ed un generico problema di programmazione lineare intera. Il progetto di tali procedure è oggi oggetto di un'intensa attività di ricerca internazionale.

Un'altra possibilità consiste nello studiare una classe specifica di problemi di programmazione lineare intera, sfruttandone la 'struttura' peculiare. Seguendo questo approccio, il progetto di un algoritmo *branch and cut* si articola nella seguenti fasi:

- individuazione delle caratteristiche strutturali del modello di programmazione lineare intera allo studio;
- traduzione delle proprietà individuate in termini di classi di disuguaglianze valide (analisi poliedrale);
- per ogni classe *C* di disuguaglianze, definizione di procedure efficienti per la risoluzione esatta del seguente problema di separazione per la classe *C*: dato x*, individuare (se esiste) una disuguaglianza $\alpha'\mathbf{x} \leq \alpha_0$ appartenente alla classe *C* e tale che $\alpha'\mathbf{x}^* > \alpha_0$.

In pratica, per ogni classe C si ha interesse ad individuare numerose disuguaglianze violate, scelte tra quelle che massimizzano il grado di violazione $\alpha'\mathbf{x}^* - \alpha_0$. Questo permette tipicamente di accelerare la convergenza complessiva dell'algoritmo *branch and cut*.

3.1.Un esempio: il problema della selezione degli indici

Illustriamo le caratteristiche principali del metodo mediante l'esempio di un problema reale che interviene nel progetto di database relazionali.
Un database relazionale si può pensare come un insieme di dati e di procedure di interrogazione (*query*) e di aggiornamento. Tipicamente la struttura dati deve essere interrogata in tempo reale, e quindi la rapidità di risposta ad una *query* è un fattore determinante nella scelta del metodo con cui organizzare le informazioni. La risposta ad una query comporta la scansione dei dati, operazione che può essere accelerata se i record vengono mantenuti ordinati (secondo una qualche chiave) mediante l'uso di uno o più indici.
Il tempo di risposta ad una determinata query è quindi funzione dell'indice utilizzato. D'altro canto, ciascun indice comporta un costo fisso relativo alle operazioni periodiche di aggiornamento dei dati, ed ha una sua occupazione di memoria.

Consideriamo un caso numerico. Vi sono m = 6 query e n = 5 indici potenziali. Vi è poi un indice fittizio, lo 0, il cui uso corrisponde in realtà alla scansione sequenziale dei dati. La seguente tabella quantifica i costi (tempi) di risposta a ciascuna query, a seconda dell'indice utilizzato; si suppone che non sia possibile utilizzare più di un indice per la stessa query:

query	Indice 0	Indice 1	Indice 2	Indice 3	Indice 4	Indice 5
1	6200	1300	6200	6200	6200	6200
2	2000	900	700	2000	2000	2000
3	800	800	800	800	800	800
4	6700	6700	6700	1700	6700	2700
5	5000	5000	5000	2200	1200	4200
6	2000	2000	2000	2000	2000	750

Il costo fisso e la dimensione (in Mbyte) degli indici sono i seguenti:

	Indice 1	Indice 2	Indice 3	Indice 4	Indice 5
costo fisso	200	1200	400	2400	250
dimensione	10	5	10	8	6

Lo spazio totale su disco a disposizione degli indici è D = 19 Mbyte.
Una soluzione ammissibile corrisponde ad un qualunque sottoinsieme di indici di dimensione totale non superiore a D. Il costo complessivo della soluzione si ottiene sommando i costi fissi e i costi relativi alle m query, secondo alcune considerazioni.
Consideriamo, ad esempio, la soluzione S = {1,5}, che occupando solo 16 Mbyte risulta

ammissibile. Tale soluzione non prevede la costruzione degli indici 2, 3 e 4 che quindi devono essere 'cancellati' dalle tabelle precedenti. Per ogni query $i \in \{1,...,m\}$ dobbiamo decidere quale indice $j \in \{0,1,5\}$ conviene selezionare. Per la query 1 converrà scegliere il primo (costo minimo 1300 sulla riga). Quest'indice sarà anche utilizzato per la query 2 (costo 900), mentre il quinto sarà assegnato alle query 4 (costo 2700), 5 (costo 4200) e 6 (costo 750). L'indice 0 è invece utilizzabile per la query 3 (costo 800). Il costo complessivo per rispondere a tutte le query è allora 10 650. A questo dobbiamo aggiungere i costi fissi (200 e 250) relativi alla costruzione/aggiornamento degli indici 1 e 5. Il costo totale della soluzione *S* è quindi 10 650 + 450 = 11 100.

Supponiamo di voler progettare un algoritmo branch and cut per questo problema. Il primo passo consiste nel definire un valido modello di programmazione lineare intera. A tal fine definiamo:

- m = numero totale di query;
- n = numero totale di potenziali indici;
- D = dimensione della memoria a disposizione per gli indici selezionati;
- c_j = costo fisso (positivo) dell'indice $j \in \{1,...,n\}$;
- d_j = dimensione (positiva) dell'indice $j \in \{1,...,n\}$;
- γ_{ij} = costo (positivo) per rispondere alla query $i \in \{1,...,m\}$ mediante l'indice $j \in \{1,...,n\}$.

Le variabili di decisione del problema sono, per ogni $j \in \{1,...,n\}$:

y_j = 1 se l'indice j viene selezionato

0 altrimenti.

Per esprimere il fatto che ogni query è associata ad un indice selezionato introduciamo inoltre la variabile

x_{ij} = 1 se la query i usa l'indice j

0 altrimenti,

definita per ogni $i \in \{1,...,m\}$ e per ogni $j \in \{1,...,n\}$.

Una possibile formulazione è allora la seguente.

$$\min \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=0}^{n} \gamma_{ij} x_{ij}$$

$$\sum_{j=1}^{n} d_j y_j \leq D \qquad (a)$$

$$(b)$$

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad i \in \{1,...,m\}$$

$$\sum_{i=1}^{m} x_{ij} \leq m y_j \qquad j \in \{1,...,n\}$$

$$x_{ij} \in \{0,1\} \qquad i \in \{1,...,m\}\ ;\ j \in \{1,...,n\}$$

$$y_j \in \{0,1\} \qquad j \in \{1,...,n\}$$

Il vincolo (*a*) garantisce che ogni query utilizzi un solo indice; il vincolo (*b*) esprime la consistenza logica dei valori x_{ij} e y_j. In particolare, quando $y_j = 0$ il vincolo impone che anche $x_{ij} = 0$ per ogni $i \in \{1,...,m\}$ (se l'indice *j* non viene selezionato, allora non può essere utilizzato da nessuna query). Per effetto del fattore m nel termine di destra, il vincolo correttamente diviene inattivo quando $y_j = 1$, dato che in questo caso l'indice *j* può essere utilizzato al limite da tutte le *m* query.

A questo punto disponiamo di una formulazione di programmazione lineare intera per il problema. Se questa si rivelerà abbastanza 'stretta', potremo verosimilmente risolvere problemi di dimensioni reali (alcune centinaia di indici e query). In caso contrario, potremmo avere difficoltà a risolvere istanze anche piccolissime, con soli 15-20 indici e query.

L'ideale sarebbe che la soluzione (**x***, **y***) del rilassamento continuo del modello avesse poche componenti frazionarie, ed in ogni caso un costo vicino al valore ottimo intero (lower bound 'stretto'). In realtà questo non succede praticamente mai, principalmente per l'errore di modellizzazione che abbiamo commesso scrivendo i vincoli. Infatti, finché manteniamo la condizione 'y_j intero' questi vincoli agiscono come richiesto. Rilassando la condizione di interezza si ha però che il modello, nel tentativo di minimizzare i costi, attribuirà a y_j^* il più piccolo valore compatibile con il vincolo (*b*), e cioè definirà sistematicamente

$$y_i^* = \frac{1}{m} \sum_{i=1}^{m} x_{ij}^*.$$

Quindi, ammettendo che i valori x_{ij} siano tutti interi si avrà $y_j^* = 1$ solo se

$$\sum_{i=1}^{m} x_{ij}^* = m$$

cioè solo nel caso, poco realistico, che un indice *j* selezionato sia poi utilizzato da tutte le *m* query. In altri termini, il coefficiente m della variabile y_j nel vincolo (*b*) fa sì che i valori y_j siano tipicamente lontani da 1.

Un primo rafforzamento dei vincoli (*b*) si ottiene come segue. Consideriamo la colonna

associata all'indice j = 1 nella tabella dei costi delle query, e notiamo che alcuni costi $\acute{Y}_{ij}$ sono uguali al corrispondente costo $\acute{Y}_{i0}$ sulla colonna 0. In questo caso si può fissare x_{ij} = 0, dato che ogni soluzione ottima non ha interesse ad usare l'indice j per la query i, potendo utilizzare con lo stesso costo l'indice 0. E' dunque lecito fissare a zero, cioè eliminare dal modello, tutte le variabili x_{ij} tali che $\acute{Y}_{ij} \geq \acute{Y}_{i0}$. Per ogni indice $j \in \{1,...,n\}$ risultano così attive le sole variabili x_{ij} con $i \in I_j$, dove $I_j = \{ i \in \{1,...,m\}: \acute{Y}_{ij} < \acute{Y}_{i0}\}$ è l'insieme delle query che hanno convenienza ad usare l'indice j.

Nell'esempio numerico, le variabili attive sono $x_{10},...,\ x_{60},\ x_{11},\ x_{21},\ x_{22},\ x_{43},\ x_{53},\ x_{54},\ x_{45},...,\ x_{65}$, oltre naturalmente alle variabili $y_1,...,y_5$.

Oltre a diminuire il numero di variabili del problema, questa riduzione permette di scrivere i vincoli (*b*) come

$$\sum_{i \in I_j} x_{ij} \leq |I_j| y_i \qquad j \in \{1,...,n\}. \qquad (c)$$

Questi sono più stringenti in quanto il coefficiente di y_j è stato diminuito da m a $|I_j|$.

Consideriamo ora il rilassamento continuo del nuovo modello, relativamente all'esempio numerico. La soluzione ottima (**x***, **y***) ha le seguenti componenti non nulle:

x^*_{20} = 6/10

x^*_{30} = 1

y^*_1 = 7/10

x^*_{11} = 1

x^*_{21} = 4/10

y^*_3 = 1

$x^*_{43} = x^*_{53}$ = 1

y^*_5 = 1/3

x^*_{65} = 1

ed un costo (arrotondato) pari a 8 940. C'è una differenza notevolissima (gap = 2 160) tra il lower bound 8 940 ed il valore ottimo intero 11 100. Applicando direttamente l'algoritmo branch and bound si generano quindi moltissimi sottoproblemi, ed il tempo di calcolo necessario diventa proibitivo.

Per ovviare a questo problema, utilizziamo l'algoritmo branch and cut.

Come primo passo, studiamo le proprietà specifiche del problema in questione, tentando di individuare nuove classi di disuguaglianze valide (analisi poliedrale). Analizzando la soluzione frazionaria dell'esempio, si nota che un 'difetto' è che l'indice 1 viene selezionato al 70 % (y^*_1 = 7/10), ma viene utilizzato al 100 % per la query 1 (x^*_{11} = 1). Sembra naturale invece imporre che un indice selezionato parzialmente sia utilizzato anche parzialmente dalle

query. Questa condizione strutturale, specifica del problema allo studio, può essere espressa dal vincolo lineare $x_{11} \leq y_1$. Abbiamo così individuato una prima classe C_1 di disuguaglianze valide per il nostro problema:

Classe C_1: $x_{ij} \leq y_j$, j ∈ {1,...,n}, $i \in I_j$.

Tali vincoli sono analoghi a (c), in quanto impongono la relazione di congruenza $y_j = 0 \rightarrow x_{ij} = 0$.

Per questa classe C_1 di disuguaglianze, dobbiamo definire una procedura efficiente per la risoluzione esatta problema di separazione per la classe. Tale problema è:

dato (**x***, **y***), individuare (se esiste) una coppia (i,j) tale che $x^*_{ij} \geq y^*_j$,

Nell'esempio numerico, la procedura individua due vincoli violati, cioè $x_{11} \leq y_1$ *e* $x_{65} \leq y_5$. Aggiungiamo questi tagli al modello corrente, e riottimizziamo con l'algoritmo del simplesso duale ottenendo una soluzione ottima di costo 9 900 (gap = 1200) e con le seguenti componenti non nulle:

$x^*_{30} = 1$

$x^*_{60} = 3/4$

$y^*_1 = 7/10$

$x^*_{11} = x^*_{21} = 1$

$y^*_3 = 3/4$

$x^*_{43} = x^*_{53} = 3/4$

$y^*_5 = 1/4$

$x^*_{45} = x^*_{55} = x^*_{65} = 1/4$.

Applicando l'algoritmo di separazione per la classe C_1 non si sono generati tagli violati, in quanto la soluzione corrente soddisfa tutti i vincoli. A questo punto possiamo effettuare il branching, oppure studiare il punto frazionario alla ricerca di una nuova classe di disuguaglianze valide.

Osserviamo che gli indici 3 e 4 non possono essere scelti contemporaneamente, dato che eccedono la memoria D disponibile. Si ha dunque la condizione $y_1 + y_3 < 3$, rafforzabile in $y_1 + y_3 \leq 1$ dato che y_1 e y_3 devono essere interi. Quest'ultima disuguaglianza è violata dal punto frazionario corrente, dato che $y^*_1 = 1$ e $y^*_3 = 3/4$. Si ha così la seguente nuova classe di vincoli validi:

$$Classe\ C_2: \quad \sum_{j \in S} y_j \leq |S| - 1,\ \forall\ S \subseteq \{1,...,n\},\ S \neq \emptyset\ : \sum_{j \in S} d_j > D. \qquad (d)$$

poiché esistono moltissimi sottoinsiemi non vuoti

$$S \subseteq \{1,...,n\}$$

la classe può contenere un numero elevatissimo di disuguaglianze. Questo è certamente un fattore positivo: più disuguaglianze ci sono, più stringente è la formulazione risultante (cioè vi sono maggiori possibilità di individuare un taglio violato). Naturalmente, l'algoritmo di separazione per la classe C_2 non può semplicemente enumerare tutti i sottoinsiemi S, verificando che

$$\sum_{j \in S} d_j > D \quad e \sum_{j \in S} y^*_j > |S| - 1:$$

il tempo di calcolo sarebbe enorme.

Un approccio più sofisticato consiste nel formulare il problema di separazione stesso come un problema di programmazione lineare intera a sé stante, per esempio mediante branch and bound. A tal fine, per ogni j ϵ {1,...,*n*} si possono introdurre le variabili ausiliarie

x_j = 1 se *j* appartiene ad S

0 altrimenti,

e riformulare il problema di separazione come quello di individuare, se esiste, un vettore **z** ϵ $\{0, 1\}^n$ tale che:

$$w = \sum_{j=1}^{n} z_j - \sum_{j=1}^{n} y^*_j z_j < 1 \quad (\rightarrow 1 - w = \textit{violazione del vincolo})$$

$$\sum_{j=1}^{n} d_j z_j \geq D + \varepsilon, \qquad \textit{validità del vincolo} \qquad (e)$$

ove ε > 0 è un valore sufficientemente piccolo (per esempio, ε = 1 se i valori d_1,..., d_n e D sono tutti interi). Volendo individuare il vincolo (*d*) più violato, si può allora interpretare (*e*) come funzione obiettivo, e scrivere il seguente modello di programmazione lineare intera per il problema di separazione:

$$w = \min \sum_{j=1}^{n} (1 - y^*_j) z_j$$

$$\sum_{j=1}^{n} d_j z_j \geq D + \varepsilon$$

$$x_j \in \{0,1\}, \quad j \in \{1,...,n\}.$$

Questo problema è noto come problema dello zaino (knapsack), già visto precedentemente.

Risolvendo questo semplice problema di programmazione lineare intera si individuerà dunque il sottoinsieme *S* cui corrisponde la massima violazione del vincolo (*d*): se w* < 1, questo vincolo è in effetti violato, e può essere aggiunto al rilassamento continuo corrente. Altrimenti, la soluzione (**x***, **y***) corrente soddisfa tutti i potenziali vincoli (*d*).

Un'altra possibilità è limitare euristicamente la scelta dei sottoinsiemi S imponendo $|S| \leq k$ per un qualche valore k fissato.

Nell'esempio numerico, applicando l'algoritmo di separazione per k = 2 si ottiene il vincolo violato $y_1 + y_3 \leq 1$. Aggiungendo questo vincolo alla formulazione corrente e riottimizzando si ottiene una soluzione ottima frazionaria di costo 10 880 (gap = 220), con le seguenti componenti non nulle:

$$x^*_{30} = 1$$
$$y^*_1 = 1 \quad x^*_{11} = x^*_{21} = 1$$
$$x^*_{54} = \frac{3}{8}$$
$$y^*_5 = 1$$
$$x^*_{45} = x^*_{65} = 1$$
$$y^*_4 = \frac{3}{8}$$
$$x^*_{55} = \frac{5}{8}.$$

La procedura di separazione per la classe C_1 non produce tagli violati, così come quella per la classe C_2 fissando k = 2. Provando con k = 3 si individua invece il sottoinsieme S = {1, 4, 5}, cui corrisponde il vincolo violato $y_1 + y_4 + y_5 \leq 2$. Aggiungendo questo vincolo e riottimizzando si ottiene una soluzione di costo 11 100 (gap = 0) e componenti non nulle

$$x^*_{30} = 1$$
$$y^*_1 = 1$$
$$x^*_{11} = x^*_{21} = 1$$
$$y^*_5 = 1$$
$$x^*_{45} = x^*_{55} = x^*_{65} = 1.$$

Poiché non esistono componenti frazionarie, questa soluzione è ottima e l'algoritmo branch and cut termina senza la necessità di effettuare il branching.

Naturalmente è necessario verificare che il procedimento descritto sia efficace per problemi di dimensioni più elevate, e su dati reali. A titolo di curiosità si riporta un confronto computazionale tra la tecnica branch and bound (senza i vincoli delle classi C_1 e C_2) e branch and cut. I risultati sono valori medi su 10 problemi reali; i tempi sono espressi in secondi di CPU e si riferiscono ad un computer SUN Sparc-2.

		Branch and Bound		Branch and Cut	
m	*n*	nodi	tempo	nodi	tempo
250	150	> 10 000	> 2310,3	1	1,3
375	225	> 10 000	> 3030,2	17	31,2
500	300	> 10 000	> 4397,0	6	32,3

L'algoritmo branch and bound non ha risolto all'ottimo nessuna delle 30 istanze considerate, avendo superato il dimensionamento massimo della cosa dei nodi (10 000 nodi). Le stesse istanze sono state invece risolte facilmente, in pochi secondi, utilizzando la tecnica branch and cut. Utilizzando tale metodo, il tempo di calcolo speso per elaborare ciascun nodo aumenta, ma il numero complessivo dei nodi si riduce drasticamente (da oltre 10 000 nodi a meno di 10, in media).

CONCLUSIONI

Abbiamo visto come il metodo più efficace per la risoluzione di problemi reali sia il branch and cut, binomio tra diversi metodi che quindi prende il meglio da entrambi.
Mettendo a confronto i tempi di risoluzione del problema pratico della risoluzione degli indici mediante branch and cut e mediante branch and bound, vediamo come nel primo caso la soluzione viene trovata rapidamente, mentre nel secondo non solo i tempi di calcolo sono lunghissimi, ma non si raggiunge addirittura nessuna soluzione ottima a causa del numero così elevato di nodi che dovrebbero essere sviluppati.
Il branch and cut è quindi un procedimento sì, se vogliamo, complesso dal punto di vista dello sviluppo, ma che in compenso si rivela estremamente efficace e veloce.
E' proprio per questo che la ricerca per il miglioramento dell'algoritmo è in continuo sviluppo, e gli esperti ricercano sempre continui modi per aumentarne l'efficacia. E' attualmente uno dei metodi più maggiormente applicati in varie situazioni reali della ricerca operativa.
In generale, la programmazione lineare intera svolge un ruolo chiave nella società moderna, andando a risolvere tutti quei problemi (e altri) di cui abbiamo analizzato gli esempi.

BIBLIOGRAFIA e SITOGRAFIA

Carlo Vercellis, aprile 2006, *Ottimizzazione – Teoria, metodi, applicazioni: Mc Graw Hill.*

Matteo Fischetti, 1999, *Lezioni di ricerca operativa,* Padova: Edizioni Libreria Progetto, seconda edizione.

Www.it.wikipedia.org/wiki

http://caronte.dma.unive.it/~basso

http://www.dis.uniroma1.it/~fasano

Fonti in lingua inglese:

http://web.mit.edu/15.053/www/AMP-Chapter-09.pdf