Metodi e Modelli per l'Ottimizzazione Combinatoria

Metodi Risolutivi per Programmazione Lineare Intera

L. De Giovanni M. Di Summa G. Zambelli

Indice

1 Definizioni preliminari

Un *problema di programmazione lineare intera* è una problema della forma

$$\begin{aligned} z_I = \max\, & c^T x \\ & Ax \leq b \\ & x \geq 0 \\ & x_i \in \mathbb{Z}, \qquad i \in I. \end{aligned} \tag{1}$$

ove $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ e $I \subseteq \{1, \ldots, n\}$ è l'insieme degli indici delle *variabili intere*. Le variabili x_i, $i \notin I$ sono invece dette *variabili continue*. Se il problema ha sia variabili intere che variabili continue, allora è detto un *problema di programmazione lineare intera mista*, mentre se tutte le variabili sono intere, il problema è detto di *programmazione lineare intera pura*.

L'insieme

$$X = \{x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0,\, x_i \in \mathbb{Z} \text{ per ogni } i \in I\}$$

è la *regione ammissibile* del problema.

$$\begin{aligned} z_L = \max\, & c^T x \\ & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{2}$$

è detto il *rilassamento lineare* di (1).

Si noti il seguente facile fatto:

$$z_I \leq z_L. \tag{3}$$

Infatti, se x^I è la soluzione ottima di (1) e x^L è la soluzione ottima di (2), allora x^I soddisfa i vincoli di (2), e dunque $z_I = c^T x^I \leq c^T x^L = z_L$.

Di seguito daremo i fondamenti di due dei metodi su cui si basano gli attuali *solver di programmazione lineare intera*: il metodo del Branch-and-Bound e il metodo dei piani di taglio.

2 Branch and Bound per programmazione lineare intera

Il metodo più comune per risolvere problemi di programmazione lineare è il metodo cosiddetto di *Branch and Bound*. Il metodo si basa sulla seguente semplice osservazione:

> *Data una partizione della regione ammissibile X in insiemi $X_1, \ldots, X_n$, sia $z_I^{(k)} = \max\{c^T x \mid x \in X_k\}$. Allora*
>
> $$z_I = \max_{k=1,\ldots,n} z_I^{(k)}.$$

Dunque il metodo di Branch and Bound procede partizionando X in sottoinsiemi più piccoli, e risolvendo il problema $\max c^T x$ su ogni sotto-insieme. Questo viene fatto ricorsivamente, dividendo a loro volta le regioni ammissibili dei sotto-problemi in sottoinsiemi. Se tale ricorsione venisse svolta completamente, alla fine enumereremmo tutte le possibili soluzioni intere del problema. Questo ovviamente presenta due problemi: prima di tutto, se il problema ha infinite soluzioni l'enumerazione completa non è possibile, e anche se la regione ammissibile contenesse un numero finito di punti, tale numero potrebbe essere esponenzialmente grande, e quindi enumerare richiederebbe un tempo eccessivo.

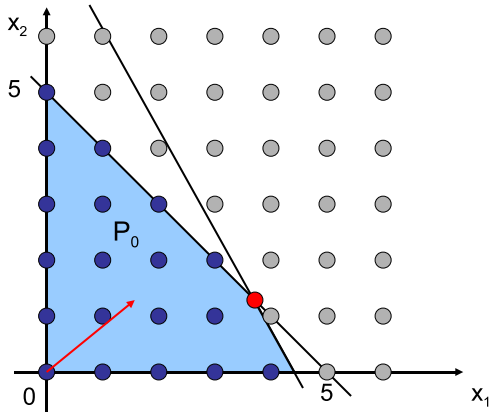
L'algoritmo di Branch and Bound cerca di esplorare solo aree "promettenti" della regione ammissibile, mantenendo upper e lower bounds al valore ottimo della soluzione in una certa area, e cercando di utilizzare tali bounds per escludere a priori certi sotto-problemi.

Di seguito, esponiamo il metodo con un esempio. I principi generali, validi per una classe di problemi di ottimizzazione combinatoria più ampia della programmazione lineare intera, saranno illustrati successivamente.

Esempio Si consideri il problema (P_0):

$$
\begin{array}{rrcll}
z_I^0 = \max & 5x_1 + \frac{17}{4}x_2 & & & \\
& x_1 + x_2 & \leq & 5 & \\
& 10x_1 + 6x_2 & \leq & 45 & (P_0) \\
& x_1, x_2 & \geq & 0 & \\
& x_1, x_2 & \in & \mathbb{Z} &
\end{array}
$$

La regione ammissibile di (P_0) e del suo rilassamento lineare è rappresentato nella figura successiva.



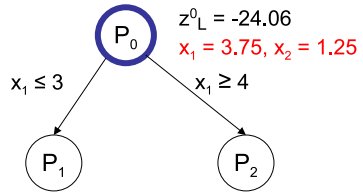
Risolvendo il rilassamento lineare di (P_0), otteniamo la soluzione ottima $x_1 = 3.75$, $x_2 = 1.25$, con valore $z_L^0 = 24.06$.

Dunque abbiamo ottenuto un upper bound per il valore ottimo z_I^0 di (P_0), ovvero $z_I^0 \leq 24.06$. Ora, poiché in una soluzione ottima di (P_0) x_1 avrà valore intero, allora la soluzione

ottima soddisferà $x_1 \leq 3$ oppure $x_1 \geq 4$. Dunque la soluzione ottima di (P_0) sarà la soluzione migliore tra le due soluzioni dei problemi (P_1) e (P_2) così definiti:

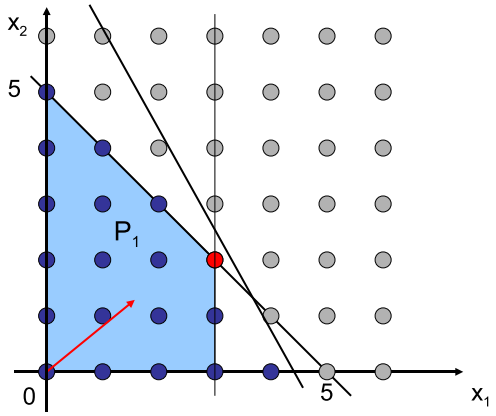
$$z_I^1 = \max \quad \begin{array}{rcl} 5x_1 + \frac{17}{4}x_2 & & \\ x_1 + x_2 & \leq & 5 \\ 10x_1 + 6x_2 & \leq & 45 \\ x_1 & \leq & 3 \\ x_1, x_2 & \geq & 0 \\ x_1, x_2 & \in & \mathbb{Z} \end{array} \qquad (P_1) \quad , \qquad z_I^2 = \max \quad \begin{array}{rcl} 5x_1 + \frac{17}{4}x_2 & & \\ x_1 + x_2 & \leq & 5 \\ 10x_1 + 6x_2 & \leq & 45 \\ x_1 & \geq & 4 \\ x_1, x_2 & \geq & 0 \\ x_1, x_2 & \in & \mathbb{Z} \end{array} \qquad (P_2)$$

Diremo che abbiamo fatto *branching sulla variabile* x_1. Si noti che in questo modo la soluzione $(3.75, 1.25)$ non appartiene al rilassamento lineare di (P_1) o (P_2). Possiamo rappresentare graficamente i sotto-problemi e i rispettivi bounds mediante un albero, detto albero di Branch-and-Bound.



I *problemi attivi* sono le foglie dell'albero, nella fattispecie (P_1) e (P_2).

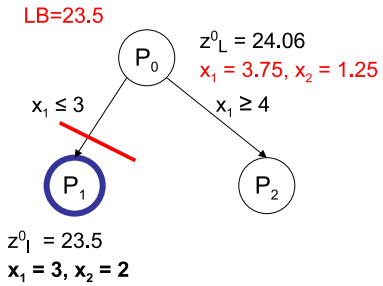
Consideriamo il problema (P_1), rappresentato in figura.



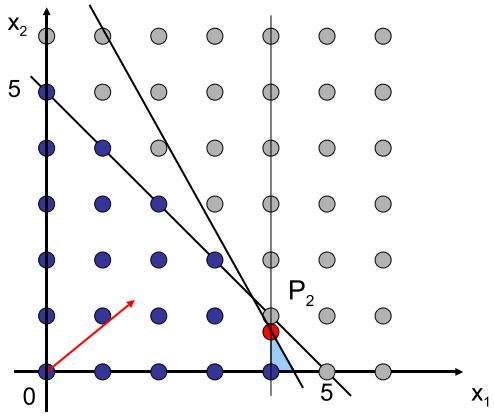
La soluzione ottima del rilassamento lineare di (P_1) è $x_1 = 3$, $x_2 = 2$, con valore $z_L^1 = 23.5$. Si noti che tale soluzione è intera, e dunque, poiché $z_I^1 \leq z_L^1$, in tal caso $(3, 2)$ è anche la soluzione ottima intera. Dunque non occorre fare ulteriore branching per il nodo (P_1), che può dunque essere potato. Diciamo che (P_1) è *potato per ottimalità*. Si noti inoltre che la soluzione ottima di (P_0) avrà valore $z_I^0 \geq z_I^1 = 23.5$. Dunque $LB = 23.5$ è un lower-bound

al valore ottimo, e $(3, 2)$ è la *soluzione intera corrente*, ovvero la miglior soluzione intera trovata fino a questo punto (detta anche *incumbent solution*).

L'albero di Branch-and-Bound è ora il seguente.



L'unica foglia non potata è $(P2)$ che è l'unico problema attivo, ed è rappresentato nella figura successiva.

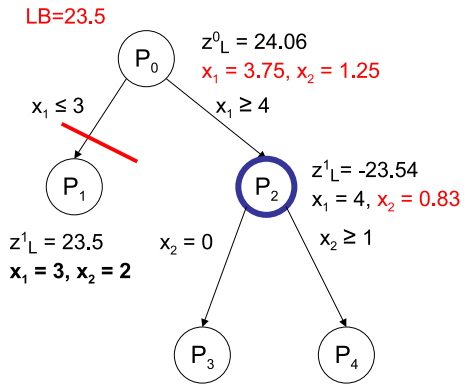


La soluzione ottima del rilassamento lineare di (P_2) è $x_1 = 4$, $x_2 = 0.83$, con valore $z_L^2 = 23.54$. Dunque $z_I^2 \leq 23.54$, e dunque 23.54 è un upper-bound al valore ottimo di (P_2). Si noti che $LB = 23.5 < 23.54$, dunque (P_1) potrebbe avere soluzione migliore della soluzione intera corrente. Poiché la componente x_2 della soluzione ottima di (P_2) ha valore 0.83, facciamo branching su x_2, ottenendo i seguenti due sotto-problemi (P_3) e (P_4) di (P_2).

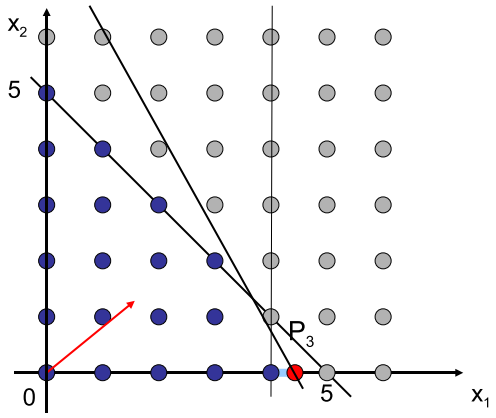
$$
\begin{array}{rrcl}
z_I^2 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& x_1 & \geq & 4 \\
& x_2 & \leq & 0 \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
\qquad (P_3) \quad ,
$$

$$
\begin{array}{rrcl}
z_I^2 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& x_1 & \geq & 4 \\
& x_2 & \geq & 1 \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
\qquad (P_4)
$$

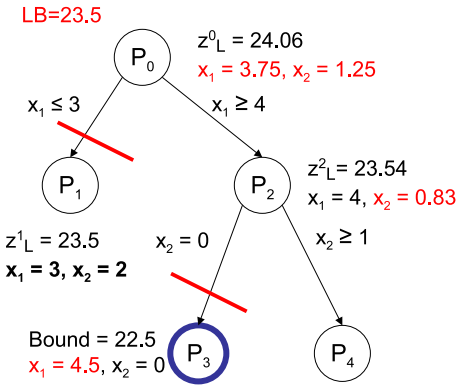
L'albero di Branch-and-Bound è ora il seguente.



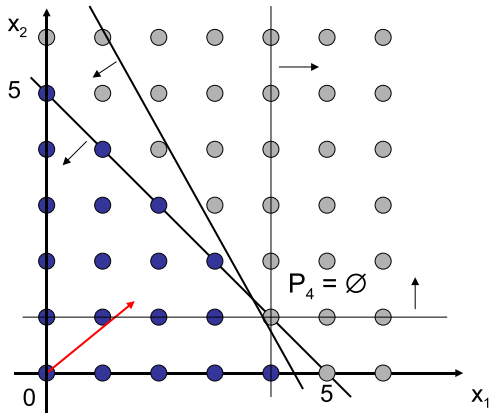
I nodi attivi sono $(P3)$ e (P_4). Risolviamo il rilassamento lineare di (P_3) (rappresentato in figura),



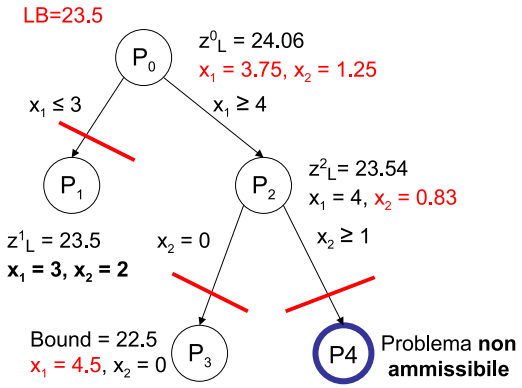
ottenendo soluzione ottima è $x_1 = 4.5$, $x_2 = 0$, con valore $z_L^3 = 22.5$. Dunque la soluzione ottima intera di (P_3) avrà valore $z_I^3 \leq 22.5$, ma poiché abbiamo già determinato una soluzione ottima intera con valore 23.5 (corrispondente a un Lower Bound), è inutile esplorare ulteriormente la regione ammissibile di (P_3) poiché sappiamo che non vi sarà nessuna soluzione intera di valore maggiore di $22.5 < 23.5$. Possiamo dunque *potare il nodo* (P_3) *per bound* (nodo *non migliorante*). L'albero di Branch-and-Bound corrente, rappresentato nella figura successiva, contiene un unico problema attivo, ovvero (P_4).



Risolvendo il rilassamento lineare di (P_4), si determina che tale rilassamento non ha alcuna soluzione ammissibile, e dunque (P_4) non ha neppure soluzioni intere, come evidente nella seguente figura.



Possiamo dunque *potare il nodo* (P_4) *per inammissibilità*. L'albero di Branch-and-Bound corrente, rappresentato nella figura successiva, non ha alcun problema attivo, e dunque la soluzione intera corrente è la migliore possibile: $(3, 2)$ è la soluzione ottima di (P_0)



Il metodo del Branch-and-Bound per programmazione lineare intera (mista) In quanto segue, esponiamo il metodo di Branch-and-Bound in maniera formale. Vogliamo risolvere il problema (P_0)

$$\begin{aligned} z_I = \max\, & c^T x \\ & Ax \leq b \\ & x \geq 0 \\ & x_i \in \mathbb{Z}, \qquad i \in I. \end{aligned}$$

ove I come al solito è l'insieme di indici delle variabili intere.

L'algoritmo manterrà un lower-bound LB sul valore ottimo z_I e manterrà l'*incumbent solution*, cioè la miglior soluzione ammissibile x^* per (P_0) trovata fino a questo punto (e dunque $x_i^* \in \mathbb{Z}$ per ogni $i \in I$, e $c^T x^* = LB$). Il metodo manterrà un albero di Branch-and-Bound $\mathcal{T}$ in cui le foglie non potate sono i *nodi attivi*. Denoteremo con ℓ l'indice massimo di un nodo (P_i) nell'albero di Branch-and-Bound.

Metodo di Branch-and-Bound

Inizializzazione: $\mathcal{T} := \{(P_0)\}$, $\ell := 0$, $LB := -\infty$, x^* non definita;

1. Se esiste un nodo attivo in $\mathcal{T}$, scegli un nodo attivo (P_k), altrimenti restituisci la soluzione ottima x^*, STOP;
2. Risolvi il rilassamento lineare di (P_k), determinando o una soluzione ottima $x^{(k)}$, di valore $z_L^{(k)}$, oppure che il rilassamento lineare di (P_k) è inammissibile.
   (a) Se il rilassamento lineare di (P_k) è inammissibile: pota (P_k) da $\mathcal{T}$ (*potatura per inammissibilità*);
   (b) Se $z_L^{(k)} \leq LB$, allora (P_k) non può avere soluzioni migliori di quella corrente x^*: pota (P_k) da $\mathcal{T}$ (*potatura per bound*);
   (c) Se $x_i^{(k)} \in \mathbb{Z}$ per ogni $i \in I$, allora $x^{(k)}$ è una soluzione ottima per (P_k) (ed ammissibile per (P_0)), dunque
      - Se $c^T x^{(k)} > LB$ (vero se non vale (b)), poni $x^* := x^{(k)}$ e $LB := c^T x^{(k)}$;
      - Pota (P_k) da $\mathcal{T}$ (*potatura per ottimialità*);
3. Se nessuno dei casi (a), (b), (c) si verifica, allora scegli un indice $h \in I$ tale che $x_h^{(k)} \notin \mathbb{Z}$, fai branching sulla variabile x_h, ponendo come figli di (P_k) in $\mathcal{T}$ i problemi

$$(P_{\ell+1}) := (P_\ell) \cap \{x_h \leq \lfloor x_h^{(k)} \rfloor\} \quad , \quad (P_{\ell+2}) := (P_\ell) \cap \{x_h \geq \lceil x_h^{(k)} \rceil\}$$

$(P_{\ell+1})$ e $(P_{\ell+2})$ sono nodi attivi, (P_k) non è attivo. Poni $\ell := \ell + 2$, torna ad 1.

Nell'algoritmo sopra riportato e nel seguito, dato un numero a, denotiamo con $\lfloor a \rfloor$ e $\lceil a \rceil$ rispettivamente l'arrotondamento per difetto e per eccesso di a.

Vi sono molti dettagli fondamentali per rendere efficiente un metodo di Branch-and-Bound. Ci soffermiamo sui seguenti aspetti.

Soluzione del rilassamento ad ogni nodo Osserviamo che il rilassamento lineare da calcolare ad un nodo corrisponde al rilassamento lineare del nodo padre con l'aggiunta di un vincolo. Se abbiamo risolto il rilassamento del nodo padre con il metodo del simplesso, abbiamo a disposizione una soluzione ottima di base per il nodo padre. Tramite una variante del metodo del simplesso chiamata "metodo del simplesso duale", si può ottenere in modo molto efficiente la soluzione ottima di un problema a cui è stato aggiunto un vincolo, a partire dalla soluzione ottima prima dell'aggiunta del vincolo stesso. Questo è uno dei fattori implementativi che accelera l'esplorazione dei vari nodi dell'albero di Branch-and-Bound per problemi di programmazione lineare intera (mista).

Scelta del nodo attivo Il passo 1 dell'algoritmo sopra esposto estrae un nodo dalla lista dei nodi attivi. Il numero di nodi che verranno aperti complessivamente dipende dalla gestione di questa lista e, in particolare, dai criteri di estrazione. Vi sono in effetti due obiettivi contrastanti che concorrono alla scelta del nodo attivo:

- trovare rapidamente una (buona) soluzione intera ammissibile. Questo ha due vantaggi: una soluzione intera fornisce un lower bound al valore ottimo del problema, e avere un buon lower bound aumenta le possiblità di potare un nodo per bound. Inoltre, qualora si decidesse di interrompere prematuramente il processo, abbiamo comunque determinato una soluzione ammissibile per il nostro problema iniziale;
- visitare il minor numero possibile di nodi.

Questi due obiettivi suggeriscono le seguenti strategie:

Depth-First-Search Questa strategia corrisponde ad una gestione LIFO (Last In First Out) della lista dei nodi attivi e ha il vantaggio che, poiché fissa le variabili l'una dopo l'altra fino a potare un nodo, tipicamente si arriva presto ad una soluzione intera. Un altro vantaggio è che vengono mantenuti pochi nodi attivi, e quindi mantenere la lista dei nodi attivi richiede poca memoria. Lo svantaggio è che si possono visitare nodi "poco promettenti", ove non vi siano soluzioni ottime o vicine all'ottimo;

Best-Node Per evitare di visitare nodi dove non vi siano buone soluzioni intere, una strategia è invece quella di scegliere un nodo attivo (P_k) con upper-bound z_L^k più grande possibile, ovvero $z_L^k = \max_t z_L^t$ ove il massimo è preso tra gli indici dei nodi attivi. In tal modo, siamo sicuri di non dividere un nodo il cui upper bound z_L^k è minore dell'ottimo valore z_I di (P_0). Uno svantaggio di questa strategia è che richiede di mantenere molti nodi attivi, e quindi richiede molta memoria.

In pratica, una strategia ibrida è di applicare inizialmente Depth-First-Search per determinare in fretta una soluzione intera, e poi passare ad una strategia di Best-Node. Inoltre, i solver di programmazione lineare intera implementano solitamente euristiche per determinare in fretta una soluzione intera, prima di iniziare il procedimento di Branch and Bound.

Valutazione di soluzioni ammissibili Per applicare efficacemente le regole di chiusura dei nodi, è necessario disporre di soluzioni ammissibili di buona qualità. Nella predisposizione di un algoritmo di Branch-and-Bound, bisogna quindi stabilire come e quando calcolare soluzioni ammissibili. Tra le varie possibilità, citiamo:

- aspettare semplicemente che l'enumerazione generi un nodo foglia ammissibile;
- implementare un algoritmo euristico che valuti una buona soluzione all'inizio, prima dell'esplorazione;
- sfruttare, con frequenza da determinare in base al tipo di problema da risolvere, l'informazione raccolta durante l'esplorazione dell'albero per costruire soluzioni ammissibili sempre migliori (ad esempio arrotondando la soluzione del rilassamento continuo in modo che risulti ammissibile).

In ogni caso, bisogna sempre valutare il compromesso tra la qualità della soluzione ammissibile corrente e lo sforzo computazionale per ottenerla.

Criteri di arresto Il metodo del Branch-and-Bound si arresta quando tutti i nodi sono dichiarati *chiusi*, cioè la lista dei nodi attivi è vuota. In questo caso, la soluzione ammissibile corrente corrisponde ad una soluzione ottima. Possono anche essere adottati criteri relativi a limiti computazionali, come ad esempio raggiunti limiti di tempo di calcolo o di memoria, ma non è garantito che l'eventuale soluzione ammissibile corrente sia ottima. Notiamo che in ogni momento durante la costruzione dell'albero di Branch-and-Bound, disponiamo non solo di un lower bound LB (dato dal valore dell'incumbent solution), ma anche di un upper bound UB, dato dal massimo di tutti i valori $z_L^{(k)}$ relativi a nodi attivi: infatti tale valore è una stima ottimistica sul valore ottimo intero z_I (nel senso che necessariamente $z_I \leq UB$). Se decidiamo di interrompere l'algoritmo anticipatamente, la differenza tra il valore dell'incumbent solution LB ed il bound UB fornisce una stima della qualità della soluzione ammissibile a disposizione. Sfruttando proprio il fatto che si ha sempre a disposizione sia un lower bound (incument solution) che un upper bound (UB), si potrebbe adottare un criterio di arresto basato sulla differenza tra questi bound: ci si ferma quando si ritiene "sufficiente" la qualità della soluzione disponibile, cioè la differenza percentuale tra upper e lower bound è sotto una soglia predefinita.

Scelta della variabile di branching Anche in questo caso, vi sono svariate strategie applicabili. Una strategia comune è quella di scegliere come variabile di branching quella

più frazionaria, cioè la cui parte frazionaria sia più vicina a $0,5$. Ovvero, posto $f_i = x_i^{(k)} - \lfloor x_i^{(k)} \rfloor$, scegliamo $h \in I$ tale che

$$h = \arg\min_{i \in I} \{\min\{f_i, 1 - f_i\}\}.$$

2.1 Branch and bound: principi generali

Il metodo del branch and bound per programmazione lineare intera mista sopra esposto è un'applicazione di uno schema più generale, valido per tutti i problemi di ottimizzazione combinatoria.

Un **problema di ottimizzazione combinatoria** è definito come:

$$\begin{array}{ll} \min / \max & f(x) \\ \text{s.t.} & x \in X \end{array}$$

dove X è un insieme *FINITO* di punti e $f(x)$ è una generica funzione obiettivo. Esempi di problemi di ottimizzazione combinatoria sono:

- problema del cammino minimo:
  $X = \{$tutti i possibili cammini da s a $t\}$, $f(x)$: costo del cammino $x \in X$.
- colorazione di un grafo:
  $X = \{$tutte le combinazioni ammissibili di colori assegnati ai vertici$\}$, $f(x)$: numero di colori utilizzati dalla combinazione $x \in X$.
- programmazione lineare:
  $X = \{$tutte le soluzioni ammissibili di base$\}$, $f(x) = c^T x$.

Un metodo generale che permette di trovare una soluzione ottima tra tutte quelle di X si basa su uno schema enumerativo che sfrutta la *finitezza* dello spazio delle soluzioni ammissibili. Lo schema è detto *Algoritmo universale per ottimizzazione combinatoria*:

1. genero tutte le possibili soluzioni x;
2. verifico l'ammissibilità della soluzione $x \in X$;
3. valuto $f(x)$
4. scelgo la x ammissibile cui corrisponde la migliore $f(x)$.

Ovviamente, lo schema è molto semplice ma soffre di due evidenti problemi. Il primo, è che la valutazione di $f(x)$ potrebbe non essere banale (ad es. potrebbe essere necessaria una simulazione per valutare la "bontà" di una soluzione). Il secondo, più generale, è che *la cardinalità di X potrebbe essere **molto** elevata.* In particolare, la seconda osservazione pone due questioni:

1. come *genero* lo spazio delle soluzioni (ammissibili)?
2. come *esploro* efficientemente lo spazio delle soluzioni?

L'algoritmo del branch-and-bound implementa lo schema dell'enumerazione sopra visto cercando di dare una risposta a queste esigenze.

Generazione delle soluzioni: operazione di branch Per capire come generare le soluzioni ammissibili, osserviamo che:

> *Dato un problema di ottimizzazione combinatoria* $z = \max / \min\{f(x) : x \in X\}$ *e data una suddivisione della regione ammissibile* X *in insiemi* $X_1, \ldots, X_n$: $\bigcup_{i=1}^{n} X_i = X$, *sia* $z^{(k)} = \max / \min\{f(x) : x \in X_k\}$. *Allora la soluzione del problema è* $z = \max / \min_{k=1,\ldots,n} z^{(k)}$.

Possiamo quindi applicare il principio *divide et impera*: si suddivide X in sottoinsiemi più piccoli, e si risolve il problema su ogni sotto-insieme. Questo viene fatto ricorsivamente, dividendo a loro volta le regioni ammissibili dei sotto-problemi in sottoinsiemi. Se tale ricorsione venisse svolta completamente, alla fine enumereremmo tutte le possibili soluzioni ammissibili del problema. La divisione ricorsiva dell'insieme delle soluzioni del problema di partenza (insieme X) dà luogo ad un *albero delle soluzioni ammissibili*, come rappresentato in Figura 1.

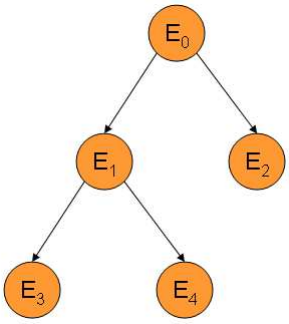


Figura 1: Generazione delle soluzioni: operazione di *Branch*.

Sia P_0 il problema di ottimizzazione combinatoria in esame cui corrisponde l'insieme delle soluzioni $E_0 = X$. E_0 è la radice dell'albero e, in generale, E_i è l'insieme delle soluzioni associate al nodo i. L'operazione di suddivisione di un nodo E_i dà luogo a dei nodi *figli*. Tale suddivisione deve garantire che

$$E_i = \bigcup_{j \text{ figlio di } i} E_j$$

cioè, le soluzioni presenti nel nodo padre devono essere presenti in (almeno) uno dei suoi figli. In altre parole, nello sviluppo dell'albero devo garantire di non perdere soluzioni

(altrimenti l'osservazione sopra riportata non sarebbe valida). Una caratteristica auspicabile, ma non necessaria, della suddivisione di un nodo E_i è la disgiunzione dei sottoinsiemi figli (la suddivisione sarebbe in questo caso una partizione in senso insiemistico):

$$E_j \cap E_k = \emptyset, \ \forall \ j,k \text{ figlio di } i$$

L'operazione di suddivisione si fermerebbe nel momento in cui ogni nodo contiene una sola soluzione. Pertanto, se E_f è un nodo foglia, si ha $|E_f| = 1$.

> La fase di costruzione di nodi figli per partizione di un nodo padre è detta *BRANCH*: un insieme di livello h viene suddiviso in t insiemi di livello $h+1$.

Esempio 1 *Si consideri un problema di ottimizzazione combinatoria con n variabili binarie $x_i \in \{0,1\}, i = 1..n$.*
In questo caso, dato un problema P_i e il corrispondente insieme di soluzioni ammissibili E_i, possiamo facilmente ottenere due sotto-problemi e due sottoinsiemi di E_i fissando una delle variabili binarie a 0 per un sotto-problema e a 1 per l'altro sotto-problema. Utilizzando questa *regola di branch* binario (ogni nodo è suddiviso in *due* nodi figli), otterremmo l'albero delle soluzioni in Figura 2.

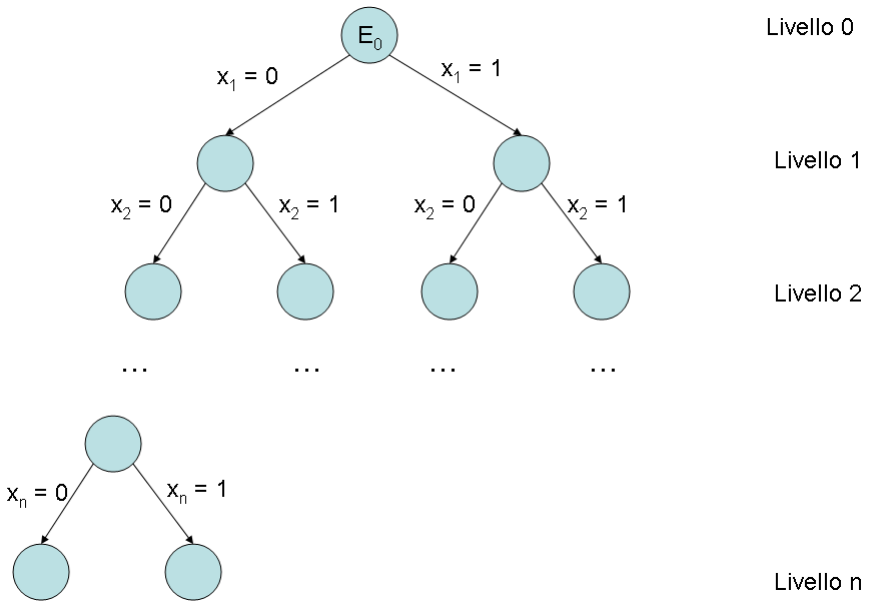


Figura 2: Branch binario.

Ad ogni livello viene fissato a 0 o a 1 il valore di una delle variabili. Un nodo di livello h, quindi, contiene tutte le soluzioni ammissibili con le variabili $x_1 \dots x_h$ fissate ad un preciso valore e, pertanto, tutti i nodi sono disgiunti. Le foglie si trovano pertanto al livello n, dove tutte le variabili sono state fissate: ogni foglia rappresenta una delle possibili stringhe binarie di n bit e, quindi, una (ed una sola) possibile soluzione del problema in questione. Si noti come il numero di foglie è 2^n e il numero di livelli è $n+1$ (incluso il livello 0 della radice).

Esplorazione efficiente: operazione di bound In generale, il numero di foglie ottenute con un'operazione di branch è esponenziale. L'esplosione completa di un albero di soluzioni, corrispondente all'enumerazione di tutte le soluzioni in X è pertanto non praticabile come metodo risolutivo. Cerchiamo quindi un metodo che ci permetta di esplorare solo aree "buone" della regione ammissibile, cercando di escludere a priori che la soluzione ottima del problema si possa trovare in altre aree. Per fare questo, consideriamo, per ogni nodo dell'albero, un *BOUND*, ossia una valutazione *ottimistica* (non peggiore) del valore che la funzione obiettivo può assumere in ciascuna delle soluzioni rappresentate dal nodo stesso. Ad esempio, consideriamo questa volta un problema di *minimizzazione* con variabili binarie e supponiamo di disporre di una soluzione ammissibile di valore 10. Supponiamo di associare, all'insieme delle soluzioni ammissibili per il problema di partenza, il nodo E_0 e di sviluppare il primo livello dell'albero di branching, fissando la variabile x_1 a 0 e a 1 e ottenendo due nodi figli E_1 ed E_2 (vedi Figura 3).

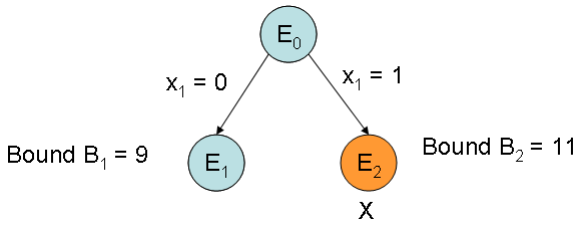


Figura 3: Uso dei Bound.

Chiamiamo $z^{(1)}$ il valore ottimo della funzione obiettivo nel sottoinsieme E_1 e $z^{(2)}$ il valore ottimo della funzione obiettivo nel sottoinsieme E_2. Ricordiamo che, come precedentemente osservato, il valore ottimo della funzione obiettivo del problema in esame è

$$z = \min\{z^{(1)}, z^{(2)}\}$$

Supponiamo ora di riuscire a stabilire, con qualche ragionamento e senza considerare ad una ad una le soluzioni, che il valore delle soluzioni ammissibili per il problema, una volta fissato a 0 il valore di x_1 non possa essere minore di 9. In altre parole, il valore della funzione obiettivo in corrispondenza di ciascuna delle soluzioni rappresentare dal nodo E_1 è sicuramente *non* inferiore a 9. 9 rappresenta quindi una valutazione ottimistica della funzione obiettivo per il sottoinsieme E_1, un limite inferiore (*lower bound*) sotto il quale il valore della funzione obiettivo non può scendere, se consideriamo solo soluzioni in E_1: cioè $z^{(1)} \geq 9$. Analogamente, supponiamo di disporre di un *lower bound* per E_2 e sia tale bound pari a 11: nessuna della soluzioni con $x_1 = 1$ (soluzioni in E_1) ha un valore della funzione obiettivo più basso di 11: cioè $z^{(2)} \geq 11$. Ora, secondo la nostra prima osservazione, $z = \min\{z^{(1)}, z^{(2)}\}$. Inoltre, utilizzando l'informazione sulla soluzione ammissibile a disposizione, $z \leq 10$ e siccome $z^{(2)} \geq 11 \geq 10$, non è possibile trovare una soluzione con valore migliore di 10 tra le soluzioni nel nodo E_2. Pertanto,

E_2 non contiene sicuramente la soluzione ottima ed è inutile sviluppare ed esplorare il sotto-albero con radice E_2.

Lo stesso ragionamento non vale per il nodo E_1: una delle soluzioni in questo nodo *potrebbe* avere valore inferiore a 10 e, pertanto, tale nodo potrebbe contenere la soluzione ottima.

In generale, se è nota una soluzione ammissibile $\bar{x}$ di valore $f(\bar{x}) = \bar{f}$, la disponibilità di un *bound* associato ai nodi dell'albero di branch ci permette di "potare" (non sviluppare) sotto-alberi che sicuramente non contengono la soluzione ottima, cioè i sotto-alberi radicati in un nodo con bound non migliore di $\bar{f}$.

Osserviamo che, anche se non abbiamo esplicitato tutti i nodi foglia del sotto-albero di E_2, siamo comunque in grado di stabilire che il valore di ciascuno di essi non è migliore di 11 e, grazie alla soluzione ammissibile, tale informazione è sufficiente per escluderli come soluzioni ottime: è come se avessimo esplorato tutto il sotto-albero in maniera *implicita*. Attraverso l'operazione di bound è quindi possibile effettuare una *enumerazione implicita* di tutte le soluzioni di un problema di ottimizzazione combinatoria.

Il metodo del Branch and Bound (B&B): schema generale Dalle osservazioni precedenti, è possibile definire il metodo del Branch and Bound (B&B) per la soluzione di problemi di ottimizzazione combinatoria. Si tratta di un metodo che enumera in modo esplicito o implicito tutte le soluzioni del problema, basandosi sui seguenti elementi:

- *operazione di branch*: costruzione dell'albero delle soluzioni ammissibili;
- disponibilità di una soluzione ammissibile di valore $\bar{f}$;
- *operazione di bound*: valutazione ottimistica della funzione obiettivo per le soluzioni rappresentate da ciascun nodo (*bound*), per evitare lo sviluppo completo di sotto-alberi (enumerazione implicita delle soluzioni rappresentate dai nodi con bound non migliore di $\bar{f}$).

Il metodo del Branch-and-Bound può essere schematizzato come segue. Dato un problema di ottimizzazione combinatoria $z = \min / \max\{f(x) : x \in X\}$, sia:

- P_0: problema di ottimizzazione iniziale;
- L: lista dei nodi aperti. Ogni nodo è una coppia (P_i, B_i), dove P_i è il sotto-problema e B_i è il relativo bound;
- $\bar{z}$: valore della migliore soluzione ammissibile.
- $\bar{x}$: migliore soluzione ammissibile corrente;

Metodo di Branch-and-Bound

0.	*Inizializzazione*:	Esegui una stima ottimistica B_0 della funzione obiettivo e poni $L = \{(P_0, B_0)\}$, $\bar{x} = \emptyset$, $\bar{z} = +\infty(\min)[-\infty(\max)]$
1.	*Criterio di Stop*:	Se $L = \emptyset$, allora STOP: $\bar{x}$ è la soluzione ottima. Se superati limiti di tempo, nodi esplorati, nodi aperti $\|L\|$ etc. STOP: $\bar{x}$ è una soluzione (non necessariamente ottima).
2.	*Selezione nodo*:	Seleziona $(P_i, B_i) \in L$ per effettuare il branch
3.	*Branching*:	Dividi P_i in t sotto-problemi $P_{ij}, j = 1..t$ $(\cup_j P_{ij} = P_i)$
4.	*Bounding*:	Valuta una stima ottimistica B_{ij} (in corrispondenza di una soluzione non necessariamente ammissibile x^R_{ij}) per ciascun sotto-problema P_{ij}
5.	*Fathoming*:	Se P_{ij} non è ammissibile, vai a 1. Se B_{ij} non è migliore di $\bar{z}$ ammissibile, vai a 1. Se x^R_{ij} è ammissibile ed è migliore di $\bar{z}$, poni $\bar{z} \leftarrow B_{ij}$, $\bar{x} \leftarrow x^R_{ij}$; elimina da L tutti i nodi k con L_k non migliore di $\bar{z}$; vai a 1. Altrimenti, aggiungi (P_{ij}, B_{ij}) a L e vai a 1.

Quello sopra esposto è uno schema di principio per la soluzione di problemi di ottimizzazione combinatoria. Per implementare un algoritmo B&B per uno specifico problema, come abbiamo fatto prima per i problemi di programmazione lineare intera mista, bisogna determinare i seguenti elementi essenziali:

(1) Regole di Branching: come costruire l'albero delle soluzioni.

(2) Calcolo del Bound: come valutare i nodi.

(3) Regole di Fathoming: come chiudere (potare) e dichiarare sondati (*fathomed*) i nodi.

(4) Regole di esplorazione dell'albero: definire le priorità di visita dei nodi aperti.

(5) Come valutare una o più soluzioni ammissibili (soluzioni da confrontare con i bound per chiudere nodi).

(6) Criteri di stop: condizioni di terminazione dell'algoritmo.

3 Formulazioni alternative

Si consideri un problema di programmazione lineare intera

$$\begin{array}{c} z_I = \max c^T x \\ Ax \leq b \\ x \geq 0 \\ x_i \in \mathbb{Z}, \qquad i \in I. \end{array} \tag{4}$$

ove $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$ e $I \subseteq \{1, \ldots, n\}$ è l'insieme degli indici delle variabili intere[1]. Sia

$$X = \{x \in \mathbb{R}^n \mid Ax \leq b,\, x \geq 0,\, x_i \in \mathbb{Z} \text{ per ogni } i \in I\}$$

la regione ammissibile del problema, e sia

$$\begin{array}{c} z_L = \max c^T x \\ Ax \leq b \\ x \geq 0 \end{array} \tag{5}$$

il rilassamento lineare di (4).

Si noti che il rilassamento lineare non è unico. Data una matrice $A' \in \mathbb{R}^{m' \times n}$ e un vettore $b' \in \mathbb{R}^{m'}$, diciamo che

$$\begin{array}{c} A'x \leq b' \\ x \geq 0 \end{array}$$

è una *formulazione* per X se vale

$$X = \{x \in \mathbb{R}^n \mid A'x \leq b',\, x \geq 0,\, x_i \in \mathbb{Z} \text{ per ogni } i \in I\}.$$

In tal caso, il problema di programmazione lineare

$$\begin{array}{c} z'_L = \max c^T x \\ A'x \leq b' \\ x \geq 0 \end{array} \tag{6}$$

è anch'esso un rilassamento lineare di (4). Chiaramente, X può avere infinite possibili formulazioni, e dunque vi sono infiniti possibili rilassamenti per (4).

[1] Da adesso in poi i coefficienti saranno considerati razionali in quanto da questa condizione dipendono molte delle proprietà di seguito illustrate, che non possono essere dimostrate se si ammettono coefficienti irrazionali. Del resto, siamo interessati a implementazioni su calcolatore degli algortmi che sfruttano tali proprietà e, di conseguenza, l'assunzione di razionalità non risulta limitativa nella pratica.

3.1 Buone formulazioni

Date due formulazioni per X,

$$Ax \leq b,\ x \geq 0$$

e

$$A'x \leq b,\ x \geq 0,$$

diciamo che la prima formulazione è migliore della seconda se

$$\{x \in \mathbb{R}^n \mid Ax \leq b,\ x \geq 0\} \subseteq \{x \in \mathbb{R}^n \mid A'x \leq b',\ x \geq 0\}.$$

La precedente nozione è giustificata dal fatto che, se $Ax \leq b$, $x \geq 0$ è una formulazione migliore di $A'x \leq b'$, $x \geq 0$, allora

$$z_I \leq z_L \leq z'_L,$$

e dunque il rilassamento lineare dato da $Ax \leq b$, $x \geq 0$ dà un bound più stretto sul valore ottimo del problema intero rispetto al rilassamento lineare determinato da $A'x \leq b'$, $x \geq 0$.

Si noti che ottenere buoni bound è fondamentale al fine di visitare pochi nodi nell'albero di Branch-and-Bound.

Esempio 2 *Facility location.*
Sono dati n possibili siti ove aprire delle *facilities* (centri che erogano un servizio/prodotto), e m clienti. Aprire la facility i, $i = 1, \ldots, n$, comporta un costo fisso f_i. Il costo di far servire il cliente j dalla facility i è c_{ij}, $i = 1, \ldots, n$, $j = 1, \ldots, m$. Ogni cliente deve essere servito da esattamente una facility. Il problema è quello di decidere quale facilities aprire e assegnare ogni cliente ad una facility che lo serva, minimizzando il costo totale.

Abbiamo le seguenti variabili decisionali:

$$y_i = \begin{cases} 1 & \text{se facility } i \text{ apre,} \\ 0 & \text{altrimenti,} \end{cases} \qquad i = 1, \ldots, n.$$

$$x_{ij} = \begin{cases} 1 & \text{se facility } i \text{ serve cliente } j, \\ 0 & \text{altrimenti} \end{cases} \qquad i = 1, \ldots, n,\ j = 1, \ldots, m.$$

Il costo totale sarà dunque dato

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

che è quindi la funzione obiettivo che dobbiamo minimizzare.

Il fatto che ogni cliente debba essere servito da esattamente una facility può essere espresso con i vincoli

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

Infine il cliente j può essere servito dalla facility i solo se i viene aperta, e dunque dobbiamo avere vincoli che esprimono il seguente fatto:

$$y_i = 0 \Rightarrow x_{ij} = 0, \quad i = 1, \ldots, n,\ j = 1, \ldots, m.$$

Questo fatto può essere espresso attraverso vincoli lineari in due modi:

Modo 1:

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

Dunque, se $y_i = 0$, allora $x_{ij} = 0$ per ogni j, mentre se $y_i = 1$, tali vincoli non pongono alcuna restrizione su x_{ij}.

Modo 2:

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

Si noti che, se $y_i = 0$, allora $x_{ij} = 0$ per ogni j, mentre se $y_i = 1$ allora il vincolo diventa $\sum_{j=1}^{m} x_{ij} \leq m$, che non pone alcuna restrizione sugli x_{ij} poiché le m è il numero totale dei clienti, e dunque alla facility i possono venire assegnati al più m clienti.

I vincoli ottenuti con il Modo 1 sono detti *vincoli non aggregati*, quelli ottenuti con il Modo 2 *vincoli aggregati*. Abbiamo dunque due possibili formulazioni per il problema di facility location.

Con i vincoli ottenuti nel primo modo otteniamo:

$$\begin{array}{l} \min \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \\ \begin{array}{rl} \sum_{i=1}^{n} x_{ij} = 1, & j = 1, \ldots, m; \\ x_{ij} \leq y_i, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq x_{ij} \leq 1, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq y_i \leq 1, & i = 1, \ldots, n; \\ x_{ij} \in \mathbb{Z},\ y_i \in \mathbb{Z} & i = 1, \ldots, n,\ j = 1, \ldots, m. \end{array} \end{array} \tag{7}$$

Con i vincoli ottenuti nel secondo modo otteniamo:

$$\begin{array}{l} \min \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \\ \begin{array}{rl} \sum_{i=1}^{n} x_{ij} = 1, & j = 1, \ldots, m; \\ \sum_{j=1}^{m} x_{ij} \leq m y_i, & i = 1, \ldots, n; \\ 0 \leq x_{ij} \leq 1, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq y_i \leq 1, & i = 1, \ldots, n; \\ x_{ij} \in \mathbb{Z},\ y_i \in \mathbb{Z} & i = 1, \ldots, n,\ j = 1, \ldots, m. \end{array} \end{array} \tag{8}$$

Si noti che la prima formulazione ha più vincoli della seconda (infatti i vincoli non aggregati sono mn, mentre i vincoli aggregati sono solo n). Tuttavia la prima formulazione è migliore della seconda. Infatti, siano P_1 l'insieme dei punti (x, y) che soddisfano

$$\begin{array}{ll} \sum_{i=1}^{n} x_{ij} = 1, & j = 1, \ldots, m; \\ x_{ij} \leq y_i, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq x_{ij} \leq 1, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq y_i \leq 1, & i = 1, \ldots, n; \end{array}$$

e P_2 l'insieme dei punti (x, y) che soddisfano

$$\begin{array}{ll} \sum_{i=1}^{n} x_{ij} = 1, & j = 1, \ldots, m; \\ \sum_{j=1}^{m} x_{ij} \leq m y_i, & i = 1, \ldots, n; \\ 0 \leq x_{ij} \leq 1, & i = 1, \ldots, n,\ j = 1, \ldots, m; \\ 0 \leq y_i \leq 1, & i = 1, \ldots, n; \end{array}$$

Dimostreremo che $P_1 \subsetneq P_2$.

Prima di tutto dimostriamo che $P_1 \subseteq P_2$. A tal fine, dimostriamo che, dato $(x, y) \in P_1$, allora $(x, y) \in P_2$. Se $(x, y) \in P_1$, allora $x_{ij} \leq y_i$ per ogni $i = 1, \ldots, n$, $j = 1, \ldots, m$. Dunque, dato $i \in \{1, \ldots, n\}$, sommando le precedenti m disuguaglianze $x_{ij} \leq y_i$, $j = 1, \ldots, m$, otteniamo $\sum_{j=1}^{m} x_{ij} \leq m y_i$, e dunque $(x, y) \in P_2$.

Infine, per far vedere che $P_1 \neq P_2$, esibiamo un punto in $P_2 \setminus P_1$. Sia $n = 2$, $m = 4$, e consideriamo il punto dato da

$$x_{11} = 1,\ x_{12} = 1,\ x_{13} = 0,\ x_{14} = 0;$$

$$x_{21} = 0,\ x_{22} = 0,\ x_{23} = 1,\ x_{24} = 1;$$

$$y_1 = \frac{1}{2},\ y_2 = \frac{1}{2}.$$

Si noti che tale punto soddisfa i vincoli aggregati ma non quelli non aggregati, poiché $1 = x_{11} \not\leq y_1 = \frac{1}{2}$. ■

Esempio 3 *Produzione multi-periodo.*
Un'azienda deve pianificare la propria produzione in un orizzonte temporale suddiviso in n periodi. Per ciascun periodo $t = 1, \ldots, n$, l'azienda ha una stima della domanda d_t di quel prodotto. Se l'azienda decide di produrre nel periodo t, essa incorre in un costo fisso f_t (indipendente dalla quantità prodotta nel periodo t) e in un costo c_t per unità prodotta. Inoltre, l'azienda può mantenere in magazzino parte di quanto già prodotto. Il costo di mantenere una unità di prodotto in magazzino dal periodo t al periodo $t + 1$ è h_t, $t = 1, \ldots, n - 1$.

L'azienda vuole decidere, quanto produrre in ciascun periodo t e quanto mettere in magazzino alla fine di ogni periodo, in modo da soddisfare la domanda in ogni periodo e di minimizzare il costo totale.

Per ogni periodo $t = 1, \ldots, n$, sia x_t la variabile che rappresenta il numero di unità prodotte nel periodo t. Sia y_t una variabile binaria che assume valore 1 se nel periodo t si produce qualcosa, 0 altrimenti. Sia infine s_t il numero di unità mantenute in magazzino dal periodo t al periodo $t+1$. Per facilitare la notazione, poniamo $s_0 = 0$.

Il costo totale è dunque

$$\sum_{t=1}^{n} c_t x_t + \sum_{t=1}^{n} f_t y_t + \sum_{t=1}^{n-1} h_t s_t$$

che è la funzione obiettivo da minimizzare.

In un certo periodo t, $t = 1, \ldots, n$, la quantità di prodotto disponibile è $s_{t-1} + x_t$. Di questa, d_t unità andranno a soddisfare la domanda per il periodo t, mentre il rimanente verà messo in magazzino fino al prossimo periodo. Dunque

$$s_{t-1} + x_t = d_t + s_t, \quad t = 1, \ldots, n.$$

Naturalmente,

$$s_t \geq 0,\ t = 1, \ldots, n-1; \quad x_t \geq 0,\ t = 1, \ldots, n.$$

Infine, dobbiamo garantire che, se produciamo in un certo periodo t, allora y_t assuma valore 1. Questo viene garantito dai vincoli seguenti:

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

ove M sia un upper bound al valore massimo di x_t. Ad esempio, $M = \sum_{t=1}^{n} d_t$. Naturalmente, dobbiamo poi imporre

$$0 \leq y_t \leq 1 \quad t = 1, \ldots, n$$

$$y_t \in \mathbb{Z} \quad t = 1, \ldots, n.$$

Si può vedere che in generale la soluzione ottima del rilassamento lineare dato da tale formulazione può avere componenti frazionarie. Ad esempio, supponiamo che i costi di magazzino siano molto elevati rispetto a tutti gli altri costi. Allora nella soluzione ottima avremo $s_t = 0$ per ogni $t = 1, \ldots, n-1$. Pertanto, per soddisfare la domanda dovremo produrre in ogni periodo esattamente la quantità necessaria a soddisfare la domanda in quel periodo. Pertanto $x_t = d_t$ $t = 1, \ldots, n$. Si noti a questo punto che, nella soluzione ottima intera, dovremmo avere $y_t = 1$, $t = 1, \ldots, n$. Tuttavia, nel rilassamento lineare possiamo porre $y_t = d_t/M < 1$.

È possibile dare una formulazione migliore, utilizzando delle variabili supplementari. Per ogni periodo i, $i = 1 \ldots, n$, e per ogni periodo $j = i, \ldots, n$, sia w_{ij} la quantità prodotta nel periodo i che viene venduta nel periodo j.

Dunque, la quantità totale prodotta per essere venduta nel periodo t sarà $\sum_{i=1}^{t} w_{it}$, pertanto dobbiamo avere

$$\sum_{i=1}^{t} w_{it} \geq d_t \quad t = 1, \ldots, n.$$

Chiaramente

$$x_t = \sum_{j=t}^{n} w_{tj} \quad t = 1, \ldots, n.$$

e

$$s_t = \sum_{i=1}^{t} \sum_{j=t+1}^{n} w_{ij} \quad t = 1, \ldots, n-1.$$

Infine, dobbiamo avere

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

Naturalmente, abbiamo sempre $y_t \in \{0, 1\}$, $t = 1, \ldots, n$. Tali vincoli forzano la variabile y_t ad 1 ogniqualvolta $x_t > 0$.

Siano P_1 l'insieme dei punti (x, y, s) che soddisfano

$$\begin{array}{cl} s_{t-1} + x_t = d_t + s_t & t = 1, \ldots, n; \\ x_t \leq M y_y & t = 1, \ldots, n; \\ x_t \geq 0 & t = 1, \ldots, n; \\ s_t \geq 0 & t = 1, \ldots, n-1; \\ 0 \leq y_t \leq 1 & t = 1, \ldots, n \end{array}$$

e P_2 l'insieme dei punti (x, y, s) per cui esiste un w tale che (x, y, s, w) soddisfi

$$\begin{array}{cl} \sum_{i=1}^{t} w_{it} \geq d_t & t = 1, \ldots, n; \\ x_t = \sum_{j=t}^{n} w_{tj} & t = 1, \ldots, n; \\ s_t = \sum_{i=1}^{t} \sum_{j=t+1}^{n} w_{ij} & t = 1, \ldots, n-1; \\ w_{ij} \leq d_j y_i, & i = 1, \ldots, n, j = i, \ldots, n; \\ w_{ij} \geq 0 & i = 1, \ldots, n, j = i, \ldots, n; \\ 0 \leq y_t \leq 1 & t = 1, \ldots, n. \end{array}$$

Non è difficile verificare che ogni punto di P_2 soddisfa i vincoli di P_1 e dunque $P_2 \subseteq P_1$. D'altra parte, il punto $s_t = 0$ $(t = 1, \ldots, n-1)$, $x_t = d_t$ (t=1,. . . , n), $y_t = d_t/M$ (t=1,. . . , n) è contenuto in P_1 ma non in P_2. Infatti, l'unico w che soddisfi i vincoli $x_t = \sum_{j=t}^{n} w_{tj}$ $(t = 1, \ldots, n)$ è $w_{tt} = d_t$ per $t = 1, \ldots, n$, $w_{ij} = 0$ per $1 \leq i < j \leq n$. Tuttavia tale punto viola il vincolo $w_{tt} \leq d_t y_t$. Pertanto $P_2 \subsetneq P_1$ ■

3.2 Formulazioni ideali

Ha quindi senso considerare la *formulazione ideale per* X, ovvero la formulazione lineare per X il cui rilassamento lineare abbia la regione ammissibile minimale (rispetto all'inclusione). In quanto segue cerchiamo di rendere formale questo concetto.

Definizione 1 *Un insieme* $P \subset \mathbb{R}^n$ *è detto un* poliedro *se esiste un sistema di vincoli* $Cx \leq d$, $x \geq 0$ *(ove* $C \in \mathbb{R}^{m \times n}$, $d \in \mathbb{R}^m$*) tale che* $P = \{x \,|\, Cx \leq d,\, x \geq 0\}$.

Diremo quindi che un poliedro P è una formulazione per l'insieme X se

$$X = \{x \in P \,|\, x_i \in \mathbb{Z}\ \forall i \in I\}.$$

Dunque, dati due poliedri P e P' che siano una formulazione per X, P sarà una formulazione migliore di P' se $P \subset P'$.

Ricordiamo che un insieme $C \subseteq \mathbb{R}^n$ è *convesso* se, per ogni coppia di punti $x, y \in C$, il segmento di retta che unisce x e y è tutto contenuto in C (Figura 4). E' facile vedere che ogni poliedro è un insieme convesso.

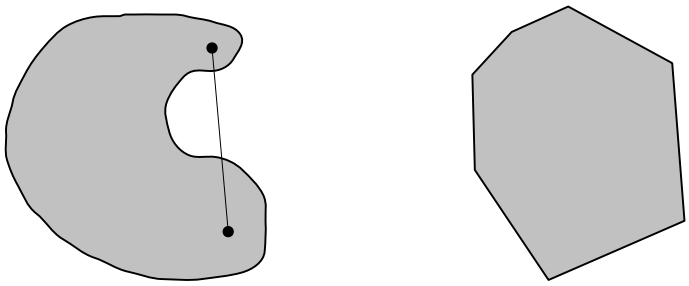


Figura 4: Un insieme non convesso ed uno convesso.

Dato un qualunque insieme $X \subseteq \mathbb{R}^n$ (nella fattispecie X è l'insieme delle soluzioni ammissibili di un problema di programmazione lineare intera) denotiamo con $\text{conv}(X)$ l'*inviluppo convesso* di X, ovvero l'insieme convesso minimale contenente X (Figura 5). In altri termini, $\text{conv}(X)$ è l'unico insieme convesso di $\mathbb{R}^n$ tale che $X \subseteq \text{conv}(X)$ e $\text{conv}(X) \subseteq C$ per ogni insieme convesso C che contenga X.

Dunque, data una formulazione $P = \{x \,|\, Cx \leq d,\, x \geq 0\}$ per X, allora, poiché P è un insieme convesso contenente X, avremo che

$$X \subseteq \text{conv}(X) \subseteq P.$$

Dunque $\text{conv}(X)$ deve essere contenuto nella regione ammissibile del rilassamento lineare di qualunque formulazione per X.

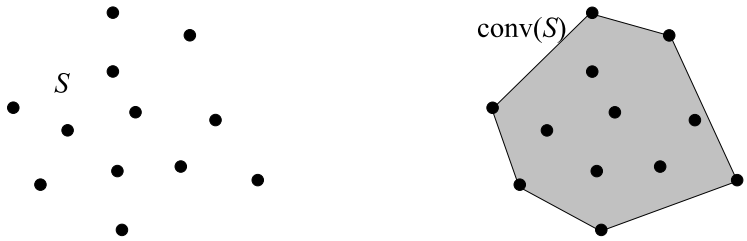


Figura 5: Un insieme ed il suo inviluppo convesso.

Il seguente è un teorema fondamentale in programmazione lineare intera, e mostra che esiste una formulazione lineare per X il cui rilassamento lineare ha come regione ammissibile esattamente $\text{conv}(X)$.

Teorema 1 (Teorema Fondamentale della Programmazione Lineare Intera) *Dati* $A \in \mathbb{Q}^{m \times n}$ *e* $b \in \mathbb{Q}^m$, *sia* $X = \{x \in \mathbb{R}^n \,|\, Ax \leq b,\, x \geq 0,\, x_i \in \mathbb{Z} \text{ per ogni } i \in I\}$. *Allora* $\text{conv}(X)$ *è un poliedro.*
Ovvero, esistono una matrice $\tilde{A} \in \mathbb{Q}^{\tilde{m} \times n}$ *e un vettore* $\tilde{b} \in \mathbb{Q}^{\tilde{m}}$ *tale che*

$$\text{conv}(X) = \{x \in \mathbb{R}^n \,|\, \tilde{A}x \leq \tilde{b},\, x \geq 0\}.$$

Dati $\tilde{A} \in \mathbb{Q}^{\tilde{m} \times n}$ e $\tilde{b} \in \mathbb{Q}^{\tilde{m}}$ tali che $\text{conv}(X) = \{x \in \mathbb{R}^n \,|\, \tilde{A}x \leq \tilde{b},\, x \geq 0\}$, diremo dunque che $\tilde{A}x \leq \tilde{b},\, x \geq 0$ è la ***formulazione ideale*** per X. Il teorema precedente dimostra che tale formulazione ideale esiste sempre.

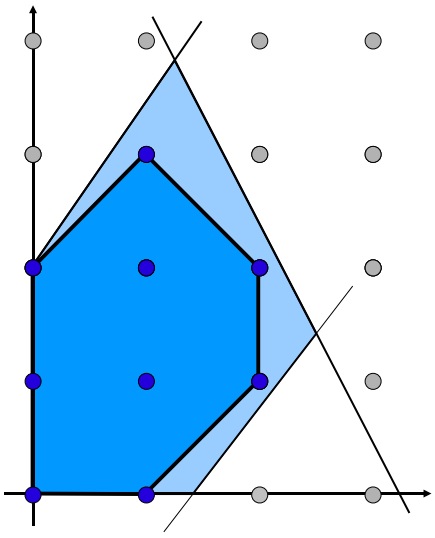


Figura 6: Una formulazione per un insieme di punti interi e la formulazione ideale.

Torniamo al problema di programmazione lineare intera

$$\begin{array}{c} z_I = \max c^T x \\ x \in X \end{array} \tag{9}$$

ove $X = \{x \in \mathbb{R}^n \,|\, Ax \leq b,\, x \geq 0,\, x_i \in \mathbb{Z} \text{ per ogni } i \in I\}$, per una certa matrice $A \in \mathbb{Q}^{m \times n}$ e vettore $b \in \mathbb{Q}^m$ dati.

Sia $\tilde{A}x \leq \tilde{b}$ la formulazione ideale per X, e consideriamo il problema di programmazione lineare

$$\begin{array}{c} \tilde{z} = \max c^T x \\ \tilde{A}x \leq \tilde{b} \\ x \geq 0. \end{array} \tag{10}$$

Aggiungendo variabili di scarto, otteniamo

$$\begin{array}{c} \tilde{z} = \max c^T x \\ \tilde{A}x + Is = \tilde{b} \\ x \geq 0,\, s \geq 0. \end{array} \tag{11}$$

Sappiamo, dalla teoria della programmazione lineare, che esiste una soluzione ottima (x^*, s^*) per (11) che sia di base per $\tilde{A}x + Is = \tilde{b}$, $x, s \geq 0$. Si noti che, per costruzione $s^* = \tilde{b} - \tilde{A}x^*$. Se $(x^*, s^* = \tilde{b} - \tilde{A}x^*)$ è una soluzione di base di $\tilde{A}x + Is = \tilde{b}$, $x, s \geq 0$, allora diremo che x^* è *una soluzione di base* per $\tilde{A}x \leq \tilde{b}$, $x \geq 0$. Dunque esiste una soluzione ottima di (10) che sia di base per $\tilde{A}x \leq \tilde{b}$, $x \geq 0$.

Teorema 2 *Sia $X = \{x \in \mathbb{R}^n \,|\, Ax \leq b,\, x \geq 0,\, x_i \in \mathbb{Z} \text{ per ogni } i \in I\}$. Il sistem $\tilde{A}x \leq \tilde{b}$, $x \geq 0$ è la formulazione ideale per X se e solo tutte le sue soluzione di base sono elementi di X. Pertanto, per ogni $c \in \mathbb{R}^n$, vale $z_I = \tilde{z}$.*

Dunque, il teorema precedente implica che risolvere il problema (10) è equivalente a risolvere (9). Infatti, data una soluzione ottima di base x^* per (10) (e dunque abbiamo $\tilde{z} = c^T x^*$), allora per il teorema precedente $x^* \in X$, e dunque x^* è una soluzione ammissibile per il problema di programmazione lineare intera (9), pertanto $\tilde{z} = c^T x^* \leq z_I$, ma poiché (10) è un rilassamento lineare di (9), vale anche $z_I \leq \tilde{z}$, e dunque $\tilde{z} = z_I$, pertanto x^* è ottima anche per (9).

Dunque, in linea di principio, risolvere un problema di programmazione lineare intera è equivalente a risolvere un problema di programmazione lineare in cui il sistema dei vincoli dia la formulazione ideale. Ci sono però due problemi, che rendono la programmazione lineare intera più difficile della programmazione lineare:

- La formulazione ideale non è nota a priori, e può essere assai difficile da determinare,
- Anche qualora fosse nota, la formulazione ideale potrebbe avere un numero assai elevato di vincoli, e dunque il problema (10) non può essere risolto direttamente con i normali algoritmi per la programmazione lineare (come ad esempio l'algoritmo del simplesso).

Esempio 4 *Matching di peso massimo.*
Sia $G = (V, E)$ un grafo non orientato. Un *matching* di G è un insieme di spigoli $M \subseteq E$ tale che gli elementi di M sono a due a due non-adiacenti. In altre parole, $M \subseteq E$ è un matching se ogni nodo di G è estremo di al più uno spigolo di G.

Il problema del matching di peso massimo è il seguente: dato un grafo non orientato $G = (V, E)$, e pesi sugli spigoli $w_e, e \in E$, determinare un matching M di G di peso totale $\sum_{e \in M} w_e$ più grande possibile.

Possiamo scrivere il problema del matching di peso massimo come un problema di programmazione lineare intera come segue. Avremo una variabile decisionale binaria x_e per ogni $e \in E$, ove

$$x_e = \begin{cases} 1 & \text{se } e \text{ è nel matching,} \\ 0 & \text{altrimenti,} \end{cases} \quad e \in E.$$

Sia $M = \{e \in \mid x_e = 1\}$. Il peso di M sarà dunque dato da

$$\sum_{e \in E} w_e x_e.$$

Affinché M sia un matching, dobbiamo garantire che, per ogni nodo $v \in V$, vi sia almeno uno spigolo $e \in E$ avente v come estremo tale che $x_e = 1$. Questo può essere espresso mediante il vincolo lineare

$$\sum_{u \in V \, t.c. \, uv \in E} x_{uv} \leq 1, \quad v \in V.$$

Dunque il problema del matching di peso massimo può essere espresso mediante il problema

$$\begin{array}{ll} \max \sum_{e \in E} w_e x_e & \\ \sum_{u \in V \, t.c. \, uv \in E} x_{uv} \leq 1, & v \in V \\ 0 \leq x_e \leq 1, & e \in E \\ x_e \in \mathbb{Z}, & e \in E. \end{array} \tag{12}$$

Si noti che tale formulazione non è ideale. Ad esempio, si consideri il caso in cui G sia un "triangolo", ovvero $V = \{a, b, c\}$ e $E = \{ab, ac, bc\}$. Siano $w_{ab} = w_{ac} = w_{bc} = 1$. Chiaramente in tale grafo tutti i matching costituiti da un unico arco hanno peso massimo, che è pari a 1. Tuttavia $x^*_{ab} = x^*_{ac} = x^*_{bc} = \frac{1}{2}$ è un soluzione ammissibile del rilassamento lineare di (12) con peso 1.5. Come si può verificare, tale soluzione è di base, quindi per il Teorema 2 questa non è la formulazione ideale.

Possiamo tuttavia ottenere una formulazione migliore della precedente aggiungendo delle disuguaglianze "ad-hoc" che possiamo ottenere guardando più attentamente la struttura del problema.

Si consideri un insieme di nodi $U \subseteq V$ contenente un numero dispari di elementi (ovvero $|U|$ dispari). Dato un qualunque matching M di G, ogni nodo in U è estremo di al più

un elemento di M, e ogni elemento di M ha al più due estremi in U. Dunque, il numero di spigoli in M con entrambi gli estremi in U è al più $|U|/2$. Poiché $|U|$ è dispari, e il numero di spigoli in M con entrambi gli estremi in U è un intero, allora M può contenere al più $(|U|-1)/2$ spigoli con entrambi gli estremi in U. Dunque, ogni punto intero x che soddisfa i vincoli di (12) deve anche soddisfare

$$\sum_{\substack{u,v\in U\\ uv\in E}} x_{uv} \leq \frac{|U|-1}{2}, \qquad \text{per ogni } U \subseteq V \text{ tale che } |U| \text{ sia dispari.}$$

Tali disuguaglianze sono dette *odd cut inequalities*.

Nell'esempio precedente del triangolo, V stesso ha cardinalità dispari, e quindi, poiché $(|V|-1)/2 = 1$, vale la disuguaglianza

$$x_{ab} + x_{ac} + x_{bc} \leq 1.$$

Si noti che il punto x^* non soddisfa tale disuguaglianza, poiché $x^*_{ab} + x^*_{ac} + x^*_{bc} = \frac{3}{2} \not\leq 1$.

Dunque la seguente è una formulazione del problema del matching di peso massimo migliore di (12)

$$\begin{array}{ll} \min \sum_{e\in E} w_e x_e & \\ \sum_{u\in V \text{ t.c. } uv\in E} x_{uv} = 1, & v \in V \\ \sum_{u,v\in U, \text{ t.c. } uv\in E} x_{uv} \leq \frac{|U|-1}{2} & U \subseteq V \ |U| \text{ dispari}, \\ 0 \leq x_e \leq 1, & e \in E \\ x_e \in \mathbb{Z}, & e \in E. \end{array} \tag{13}$$

In effetti, è possibile dimostrare che (13) è la formulazione ideale per il problema del matching di peso massimo, e che dunque sarebbe sufficiente risolvere il rilassamento lineare di tale formulazione per ottenere l'ottimo del problema intero. Naturalmente, in questo caso il rilassamento lineare ha un numero esponenziale di vincoli (infatti ci sono $2^{|V|-1}$ sottoinsiemi di V di cardinalità dispari) ed è dunque impossibile dal punto di vista pratico risolvere il problema con tutte le odd-cut inequalities (si pensi che per un grafo con soli 40 nodi vi sarebbero già più di 500 miliardi di odd-cut inequalities). Una strategia migliore è quella di risolvere una sequenza di rilassamenti lineari, a partire dal problema (12), aggiungendo una o più odd-cut inequalities alla volta che taglino la soluzione ottima corrente, fino a che non si trovi una soluzione ottima intera.

4 Il metodo dei piani di taglio

L'idea del metodo dei piani di taglio è quella di risolvere una serie di rilassamenti lineari che approssimino via via sempre meglio l'inviluppo convesso della regione ammissibile intorno alla soluzione ottima.

Più formalmente, vogliamo come al solito risolvere il problema di programmazione lineare intera (P_I)

$$\begin{array}{l}\max c^T x \\ x \in X\end{array} \qquad (P_I)$$

ove $X = \{x \in \mathbb{R}^n \,|\, Ax \leq b,\, x \geq 0,\, x_i \in \mathbb{Z} \text{ per ogni } i \in I\}$, per una certa matrice $A \in \mathbb{Q}^{m \times n}$ e vettore $b \in \mathbb{Q}^m$ dati.

Diremo che una disuguaglianza lineare $\alpha^T x \leq \beta$, ove $\alpha \in \mathbb{R}^n$ e $\beta \in \mathbb{R}$, è *valida* per X se, per ogni $x \in X$, $\alpha^T x \leq \beta$ è soddisfatta. Si noti dunque che, se $A'x \leq b'$, $x \geq 0$, è una qualunque formulazione per X, allora anche il sistema ottenuto aggiungendo una disuguaglianza per valida $\alpha^T x \leq \beta$ è una formulazione per X.

Dato $x^* \notin \text{conv}(X)$, diremo che una disuguaglianza valida per X $\alpha^T x \leq \beta$ *taglia* (o *separa*) x^* se $\alpha x^* > \beta$. Dunque, se $A'x \leq b'$, $x \geq 0$ è una qualunque formulazione per X, e x^* è un punto che soddisfa $A'x^* \leq b'$, $x^* \geq 0$, allora data una qualunque disuguaglianza $\alpha^T x \leq \beta$ valida per X che tagli x^*, anche il sistema $A'x \leq b'$, $\alpha^T x \leq \beta$, $x \geq 0$ è una formulazione per X.

Metodo dei piani di taglio

Considera come rilassamento lineare iniziale $\max\{c^T x \,|\, Ax \leq b,\, x \geq 0\}$.

1. Risolvi il rilassamento lineare corrente, e sia x^* la soluzione ottima di base;
2. Se $x^* \in X$, allora x^* è una soluzione ottima di (P_I), STOP.
3. Altrimenti, determina una disuguaglianza $\alpha^T x \leq \beta$ valida per X che tagli x^*;
4. Aggiungi il vincolo $\alpha^T x \leq \beta$ al rilassamento lineare corrente e torna ad 1.

Naturalmente, il metodo dei piani di taglio descritto è un modo generale per affrontare problemi di programmazione lineare intera, ma per implementarlo è necessario fornire un modo automatico per determinare disuguaglianze che taglino la soluzione ottima corrente. Di seguito ne esponiamo uno.

4.1 Tagli di Gomory

Il metodo dei tagli di Gomory è applicabile solo a problemi di programmazione lineare intera pura. Ovvero, consideriamo il problema

$$\begin{aligned} z_I = \min\, & c^T x \\ & Ax = b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned} \tag{14}$$

ove $A \in \mathbb{Q}^{m \times n}$ e vettore $b \in \mathbb{Q}^m$ dati. Sia $X = \{x \mid Ax = b,\, x \geq 0,\, x \in Z^n\}$.

Si risolva il rilassamento lineare di tale problema con il metodo del simplesso, ottenendo alla fine il problema in forma tableau rispetto alla base ottima B (ove con F denotiamo l'insieme di indici delle variabili fuori base)

$$\begin{array}{lrlcll} \min & z & & & & \\ & -z & + \sum_{j \in F} \bar{c}_j x_j & = & -z_B & \\ & x_{\beta[i]} & + \sum_{j \in F} \bar{a}_{ij} x_j & = & \bar{b}_i, & i = 1, \ldots, m \\ & & x & \geq & 0. & \end{array}$$

Poiché B è una base ottima, i costi ridotti sono tutti non-negativi, ovvero $\bar{c}_j \geq 0$, $j \in F$. La soluzione ottima x^* di base rispetto alla base B è data da

$$\begin{aligned} x^*_{\beta[i]} &= \bar{b}_i, \quad i = 1, \ldots, m; \\ x^*_j &= 0, \quad j \in F; \end{aligned}$$

pertanto $x^* \in \mathbb{Z}^n$ se e solo se $\bar{b}_i \in \mathbb{Z}$ per $i = 1, \ldots, m$.

Se ciò non avviene, sia $h \in \{1, \ldots, m\}$ un indice tale che $\bar{b}_h \notin \mathbb{Z}$.

Si noti che ogni vettore x che soddisfa $Ax = b$, $x \geq 0$, soddisfa anche

$$x_{\beta[h]} + \sum_{j \in F} \lfloor \bar{a}_{hj} \rfloor x_j \leq \bar{b}_h$$

poiché

$$\bar{b}_h = x_{\beta[h]} + \sum_{j \in F} \bar{a}_{hj} x_j \geq x_{\beta[h]} + \sum_{j \in F} \lfloor \bar{a}_{hj} \rfloor x_j$$

ove la disuguaglianza vale poiché $x_j \geq 0$ e $\bar{a}_{hj} \geq \lfloor \bar{a}_{hj} \rfloor$ per ogni j.

Ora, poiché stiamo risolvendo un problema lineare intero puro, ogni soluzione intera ammissibile di $Ax = b$, $x \geq 0$ soddisfa

$$x_{\beta[h]} + \sum_{j \in F} \lfloor \bar{a}_{hj} \rfloor x_j \leq \lfloor \bar{b}_h \rfloor \tag{15}$$

poiché $x_{\beta[h]} + \sum_{j \in F} \lfloor \bar{a}_{hj} \rfloor x_j$ è un numero intero.

La disuguaglianza (15) è detta *taglio di Gomory*. Dalla precedente discussione, il taglio di Gomory (15) è valido per X, e inoltre si può vedere che taglia la soluzione ottima corrente x^*, poiché

$$x^*_{\beta[h]} + \sum_{j \in F} \lfloor \bar{a}_{hj} \rfloor x^*_j = x^*_{\beta[h]} = \bar{b}_h > \lfloor \bar{b}_h \rfloor$$

ove il fatto che $\bar{b}_h > \lfloor \bar{b}_h \rfloor$ discende dal fatto che $\bar{b}_h$ non è intero.

Risulta conveniente scrivere il taglio di Gomory (15) in forma equivalente. Aggiungendo una variabile di scarto s, il taglio di Gomory (15) diventa

$$x_{\beta[h]} + \sum_{j \in F} \lfloor \bar{a}_{hj} \rfloor x_j + s = \lfloor \bar{b}_h \rfloor, \quad s \geq 0.$$

Si noti che, poiché tutti i coefficienti nell'equazione ottenuta sono interi, allora se x è un vettore con componenti intere, allora anche s deve essere intero. Possiamo dunque richiedere che s sia anch'essa una variabile intera.

Ora, sottraendo alla precedente l'equazione del tableau

$$x_{\beta[h]} + \sum_{j \in F} \bar{a}_{hj} x_j = \bar{b}_h,$$

si ottiene

$$\sum_{j \in F} (\lfloor \bar{a}_{hj} \rfloor - a_{hj}) x_j + s = \lfloor \bar{b}_h \rfloor - b_h.$$

Quest'ultima disuguaglianza è detto *taglio di Gomory in forma frazionaria*.

Aggiungendo tale vincolo al tableau ottimo precedente, otteniamo

$$\begin{array}{lrlll} \min & z & & & \\ & -z & + \sum_{j \in F} \bar{c}_j x_j & = & -z_B & \\ & x_{\beta[i]} & + \sum_{j \in F} \bar{a}_{ij} x_j & = & \bar{b}_i, & i = 1, \ldots, m \\ & & \sum_{j \in F} (\lfloor \bar{a}_{hj} \rfloor - a_{hj}) x_j + s & = & \lfloor \bar{b}_h \rfloor - b_h & \\ & & x, \quad s & \geq & 0. & \end{array}$$

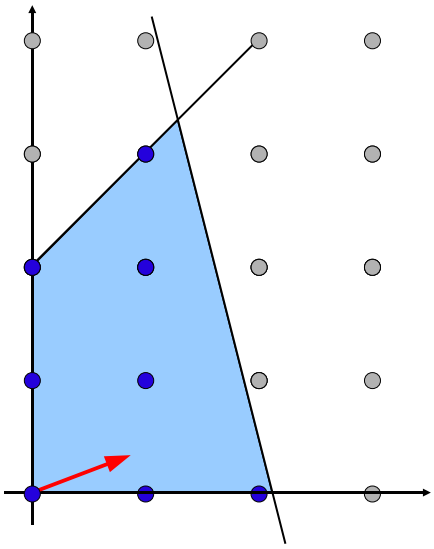
Si noti che il problema è già in forma tableau rispetto alla base in cui le variabili in base siano $x_{\beta[1]}, \ldots, x_{\beta[m]}, s$, e che tale base è ammissibile nel duale poiché i costi ridotti sono non-negativi (visto che sono i costi ridotti del tableau ottimo del problema precedente, senza il nuovo vincolo).

Si noti inoltre che $\bar{b}_i \geq 0$, $i = 1, \ldots, m$, mentre il termine noto del vincolo in cui s compare è $\lfloor \bar{b}_h \rfloor - b_h < 0$. Possiamo dunque risolvere il problema con il metodo del simplesso duale, ed alla prima iterazione dovremo far uscire la variabile s.

Esempio 5 *Si risolva il seguente problema con il metodo dei piani di taglio di Gomory.*

$$\begin{array}{rcrcl} \min\ z & = & -11x_1 - 4.2x_2 & & \\ & & -x_1 + x_2 & \leq & 2 \\ & & 8x_1 + 2x_2 & \leq & 17 \\ & & x_1, x_2 & \geq & 0 \text{ intere.} \end{array} \tag{16}$$

La regione ammissibile del relativo rilassamento lineare è rappresentata nella seguente figura:



Aggiungiamo variabili di scarto x_3 e x_4 per portare il problema in forma standard:

$$\begin{array}{rcrcl} -z - 11x_1 - 4.2x_2 & = & 0 \\ -x_1 + x_2 + x_3 & = & 2 \\ 8x_1 + 2x_2 + x_4 & = & 17 \\ x_1, x_2, x_3, x_4 & \geq & 0 \text{ intere.} \end{array}$$

Risolvendo il rilassamento lineare, otteniamo il tableau:

$$\begin{array}{rrrrcl} -z & & +1.16x_3 & +1.52x_4 & = & 28.16 \\ & x_2 & +0.8x_3 & +0.1x_4 & = & 3.3 \\ x_1 & & -0.2x_3 & +0.1x_4 & = & 1.3 \end{array}$$

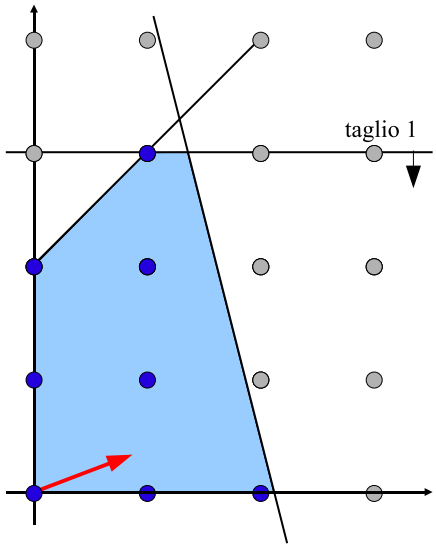
La soluzione di base corrispondente è $x_3 = x_4 = 0$, $x_1 = 1.3$, $x_2 = 3.3$ con valore obiettivo $z = -28.16$. Poiché i valori di x_1 e x_2 non sono interi, questa non è una soluzione ammissibile di (16). Possiamo ottenere un taglio di Gomory dall'equazione $x_2 + 0.8x_3 + 0.1x_4 = 3.3$, ottenendo

$$x_2 \leq 3$$

Aggiungendo questo taglio al rilassamento lineare originario, otteniamo una nuova formulazione.

$$\begin{aligned}\max\ & 11x_1 + 4.2x_2 \\ & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x_2 \leq 3 \\ & x_1, x_2 \geq 0.\end{aligned}$$

la cui regione ammissibile è la seguente



Per risolvere questo problema, otteniamo il taglio di Gomory in forma frazionaria, con variabile di scarto x_5, ovvero

$$-0.8x_3 - 0.1x_4 + x_5 = -0.3.$$

Aggiungendolo al tableau ottimo precedente

$$\begin{array}{rrrrrcr} -z & & +1.16x_3 & +1.52x_4 & & = & 28.16 \\ & x_2 & +0.8x_3 & +0.1x_4 & & = & 3.3 \\ x_1 & & -0.2x_3 & +0.1x_4 & & = & 1.3 \\ & & -0.8x_3 & -0.1x_4 & +x_5 & = & -0.3 \end{array}$$

Risolvendo con il simplesso duale, facciamo uscire di base x_5, mentre entra in base x_3, poiché $\min\{1.16/0.8, 1.52/0.1\} = 1.16/0.8$.

Otteniamo il tableau ottimo

$$\begin{array}{rrrrcr} -z & & +1.375x_4 & +1.45x_5 & = & 27.725 \\ & x_2 & & +x_5 & = & 3 \\ x_1 & & +0.125x_4 & -0.25x_5 & = & 1.375 \\ & x_3 & +0.125x_4 & -1.25x_5 & = & 0.375 \end{array}$$

La soluzione di base è $x_1 = 1.375$, $x_2 = 3$ $x_3 = 0.375$, con valore $z = 27.725$.

Dall'equazione $x_3 + 0.125x_4 - 1.25x_5 = 0.375$ del tableau, otteniamo il taglio di Gomory

$$x_3 - 2x_5 \leq 0$$

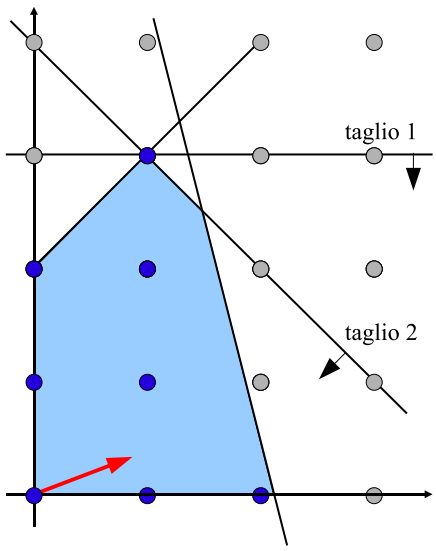
che, poiché $x_3 = 2 + x_1 - x_2$ e $x_5 = 3 - x_2$, nello spazio (x_1, x_2) è espresso da

$$x_1 + x_2 \leq 4.$$

Aggiungendo questo vincolo al problema originale, otteniamo il nuovo rilassamento lineare

$$\begin{aligned}
\max\ & 11x_1 + 4.2x_2 \\
& -x_1 + x_2 \leq 2 \\
& 8x_1 + 2x_2 \leq 17 \\
& x_2 \leq 3 \\
& x_1 + x_2 \leq 4 \\
& x_1, x_2 \geq 0.
\end{aligned}$$

rappresentato nella seguente figura



Il taglio di Gomory in forma frazionaria è

$$-0.125x_4 - 0.75x_5 + x_6 = -0.375$$

Aggiungendo il taglio al tableau ottimo precedente, otteniamo

$$\begin{array}{rrrrrrr}
-z & & +1.375x_4 & +1.45x_5 & & = & 27.725 \\
& x_2 & & +x_5 & & = & 3 \\
x_1 & & +0.125x_4 & -0.25x_5 & & = & 1.375 \\
& x_3 & +0.125x_4 & -1.25x_5 & & = & 0.375 \\
& & -0.125x_4 & -0.75x_5 & +x_6 & = & -0.375
\end{array}$$

Dobbiamo far uscire di base x_6, e far entrare in base x_5, ottenendo il tableau ottimo

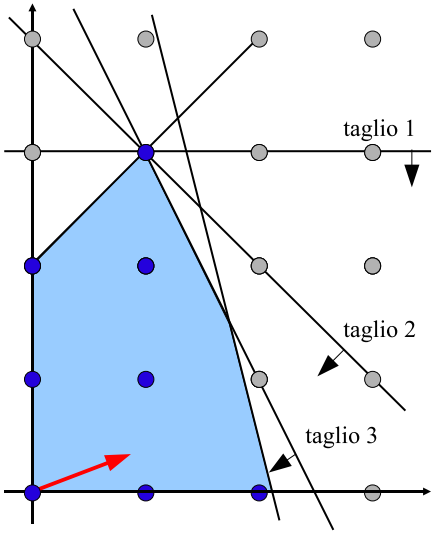
$$\begin{array}{llllll} -z & & +17/15x_4 & & +29/15x_6 & = 27 \\ & x_2 & -1/6x_4 & & +4/3x_6 & = 2.5 \\ x_1 & & +1/6x_4 & & -1/3x_6 & = 1.5 \\ & x_3 & & & +x_6 & = 0 \\ & & 1/6x_4 & +x_5 & -4/3x_6 & = 0.5 \end{array}$$

Dunque la soluzione ottima è $x_1 = 1.5$, $x_2 = 2.5$ con valore $z = 27$.

Dall'equazione $1/6x_4 + x_5 - 4/3x_6 = 0.5$ del tableau, otteniamo il taglio di Gomory $x_5 - 2x_6 \leq 0$ che, poiché $x_5 = 3 - x_2$ e $x_6 = 4 - x_1 - x_2$, nello spazio (x_1, x_2) è espresso da $2x_1 + x_2 \leq 5$. Il nuovo rilassamento lineare è

$$\begin{aligned} \max\ & 11x_1 + 4.2x_2 \\ & -x_1 + x_2 \leq 2 \\ & 8x_1 + 2x_2 \leq 17 \\ & x_2 \leq 3 \\ & x_1 + x_2 \leq 4 \\ & 2x_1 + x_2 \leq 5 \\ & x_1, x_2 \geq 0. \end{aligned}$$

in figura



Il taglio di Gomory in forma frazionaria è

$$-1/6x_4 - 2/3x_6 + x_7 = -0.5$$

Per risolvere il nuovo rilassamento, aggiungiamo il taglio al tableau ottimo precedente, ottenendo

$$\begin{array}{rrrrrrrrrrrrr}
-z & & & & +17/15x_4 & & +29/15x_6 & & & = & 27 \\
 & & x_2 & & -1/6x_4 & & +4/3x_6 & & & = & 2.5 \\
 & x_1 & & & +1/6x_4 & & -1/3x_6 & & & = & 1.5 \\
 & & & x_3 & & & +x_6 & & & = & 0 \\
 & & & & 1/6x_4 & +x_5 & -4/3x_6 & & & = & 0.5 \\
 & & & & -1/6x_4 & & -2/3x_6 & & +x_7 & = & -0.5
\end{array}$$

Dobbiamo fare due iterazioni dell'algoritmo del simplesso duale per arrivare all'ottimo: nella prima esce di base x_7 ed entra x_6, nella seconda esce x_3 ed entra x_4, ottenendo così il tableau ottimo:

$$\begin{array}{rrrrrrrrrrrrr}
-z & & & & +13/15x_3 & & & & +76/15x_7 & = & 23,6 \\
 & & x_2 & & +2/3x_3 & & & & +1/3x_7 & = & 3 \\
 & x_1 & & & -1/3x_3 & & & & +1/3x_7 & = & 1 \\
 & & & & 4/3x_3 & +x_4 & & & -10/3x_7 & = & 3 \\
 & & & & -2/3x_3 & & +x_5 & & -1/3x_7 & = & 0 \\
 & & & & -1/3x_4 & & & x_6 & -2/3x_7 & = & 0
\end{array}$$

Dunque la soluzione ottima è $x_1 = 1$, $x_2 = 3$ con valore $z = 23.6$. Questa è dunque la soluzione ottima intera del problema.