Metodi e Modelli per l'Ottimizzazione Combinatoria

Metodi Risolutivi per la Programmazione Lineare Intera - Parte I

L. De Giovanni G. Zambelli

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 $I \subseteq \{1, \dots, n\}$ 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 $z_I = c^T x^I \leq c^T x^L = z_L$.

1 Branch and Bound

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 $X_1, \dots, X_n$, sia $z_I^{(k)} = \max\{c^T x \mid x \in X_k\}$. Allora

$$z_I = \max_{k=1,\dots,n} z_I^{(k)}.$$

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.

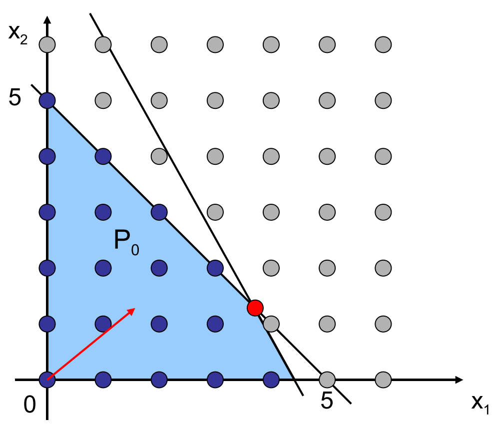
Esponiamo il metodo con un esempio.

Esempio.

Si consideri il problema (P_0):

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

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

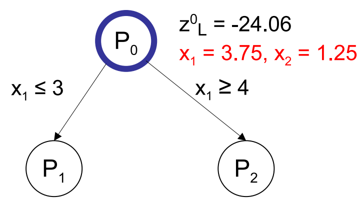


Risolvendo il rilassamento lineare di (P_0), otteniamo la soluzione ottima $x_1 = 3.75$, $x_2 = 1.75$, con valore $z_L^0 = 24.06$.

Dunque abbiamo ottenuto un upper bound per il valore ottimo z_I^0 di (P_0), ovvero $z_I^0 \leq 24. - 6$. 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:

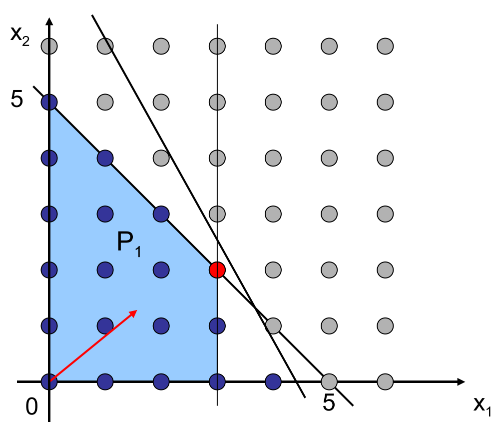
$$
\begin{array}{rrcl}
z_I^1 = \max & 5x_1 + \frac{17}{4}x_2 & & \\
& x_1 + x_2 & \leq & 5 \\
& 10x_1 + 6x_2 & \leq & 45 \\
& x_1, & \leq & 3 \\
& x_1, x_2 & \geq & 0 \\
& x_1, x_2 & \in & \mathbb{Z}
\end{array}
\quad (P_1) \quad , \quad
\begin{array}{rrcl}
z_I^2 = \max & 5x_1 + \frac{17}{4}x_2 & & \\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

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.



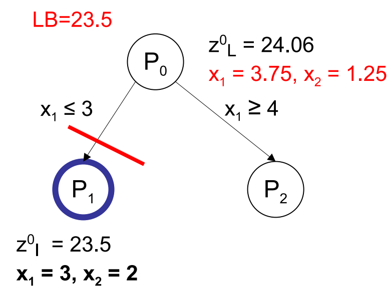
I *problemi attivi* sono le foglie dell'albero, nella fattispecie (P_1) e (P_2).

Consideriamo il problema (P_1), rappresentato in figura.

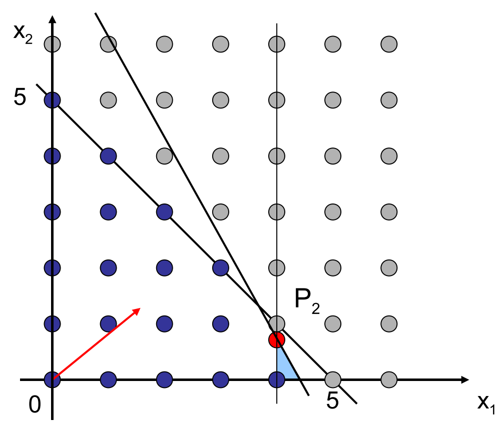


La soluzione ottima del rilassamento lineare di (P_1) è $x_1 = 3$, $x_2 = 2$, con valore $z_L^1 = 23.5$. Si noti che tale soluzione è intera, e dunque, poiché $z_I^1 \leq z_L^1$, in tal caso $(3, 2)$ è anche la soluzione ottima intera. Dunque non occorre fare ulteriore branching per il nodo (P_1), che 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.

L'albero di Branch-and-Bound è ora il seguente. L'unica foglia non potata è $(P2)$ che



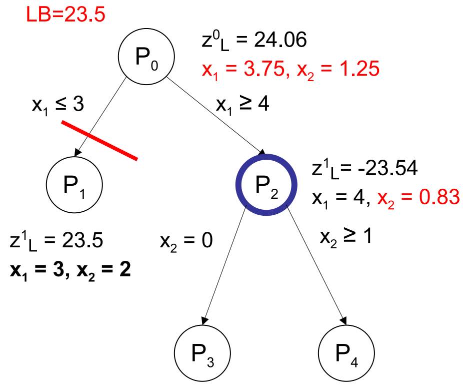
è l'unico problema attivo, ed è rappresentato nella figura successiva.



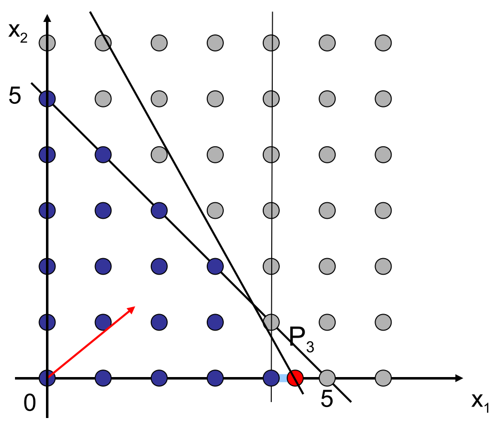
La soluzione ottima del rilassamento lineare di (P_2) è $x_1 = 4$, $x_2 = 0.83$, con valore $z_L^2 = 23.54$. Dunque $z_I^2 \leq 23.54$, e dunque 23.54 è un upper-bound al valore ottimo di (P_2). Si noti che $LB = 23.5 < 23.54$, dunque (P_1) potrebbe avere soluzione migliore della soluzione intera corrente. 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).

$$
z_I^2 = \max \quad 5x_1 + \tfrac{17}{4}x_2 \qquad\qquad\qquad\qquad\qquad z_I^2 = \max \quad 5x_1 + \tfrac{17}{4}x_2
$$

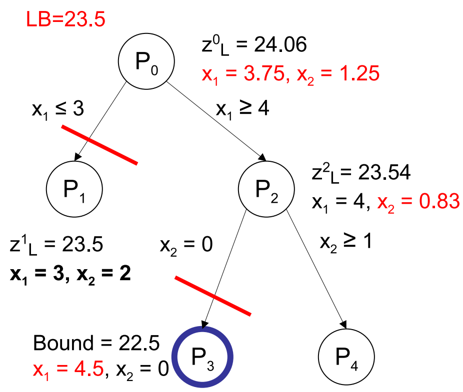
$$
\begin{array}{rcl}
x_1 + x_2 & \leq & 5 \\
10x_1 + 6x_2 & \leq & 45 \\
x_1, & \geq & 4 \\
x_2, & \leq & 0 \\
x_1, x_2 & \geq & 0 \\
x_1, x_2 & \in & \mathbb{Z}
\end{array} \qquad (P_3) \quad , \qquad
\begin{array}{rcl}
x_1 + x_2 & \leq & 5 \\
10x_1 + 6x_2 & \leq & 45 \\
x_1, & \geq & 4 \\
x_2, & \geq & 1 \\
x_1, x_2 & \geq & 0 \\
x_1, x_2 & \in & \mathbb{Z}
\end{array} \qquad (P_4)
$$



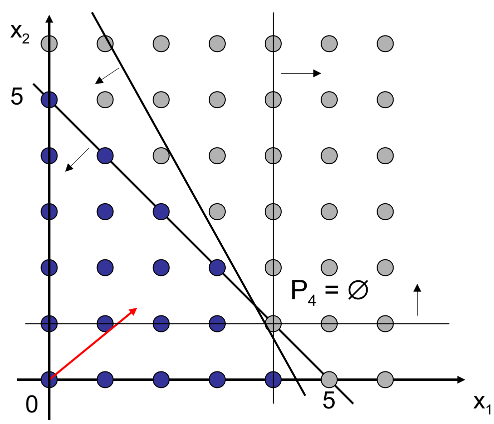
L'albero di Branch-and-Bound è ora il seguente.
I nodi attivi sono $(P3)$ e (P_4). Risolviamo il rilassamento lineare di (P_3) (rappresentato in figura),



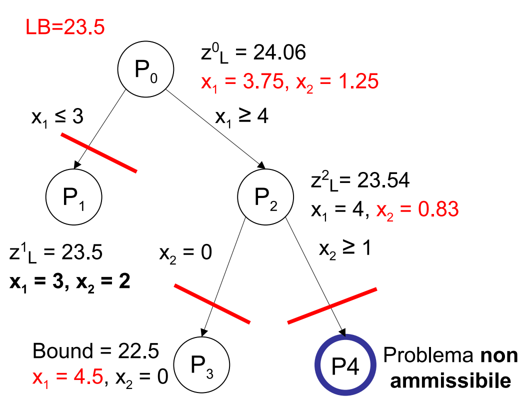
ottenendo soluzione ottima è $x_1 = 4.5$, $x_2 = 0$, con valore $z_L^3 = 22.5$. Dunque la soluzione ottima intera di (P_3) avrà valore $z_I^3 \leq 22.5$, ma poiché abbiamo già determinato una soluzione ottima intera con valore 23.5, è inutile esplorare ulteriormente la regione ammissibile di (P_3) poiché 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*. 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.



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. Dunque $(3, 2)$ è la soluzione ottima di (P_0)



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 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 ritorna 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 $c^T x^{(k)} > LB$, 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.

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

Scelta del nodo attivo Vi sono 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 possiblitá 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 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 a numeri interi implementano solitamente euristiche per determinare in fretta una soluzione intera, prima di iniziare il procedimento di Branch and Bound.

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*, cioé la cui parte frazionaria sia piú vicina a $0,5$. Ovvero, posto[1] $f_i = x_i^{(k)} - \lfloor x_i^{(k)} \rfloor$, scegliamo $h \in I$ tale che

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

[1]Dato un numero a denotiamo con $\lfloor a \rfloor$ l'arrotondamento per difetto di a, ovvero $\lfloor a \rfloor$ è l'intero minore o uguale ad a più vicino ad a stesso.