Risoluzione di problemi di programmazione lineare tramite generazione di colonne

A. Agnetis*

1 Introduzione

In alcune applicazioni, un problema può essere formulato in termini di programmazione lineare, ma con un numero di variabili talmente elevato che non è pensabile generarle tutte esplicitamente. Nel seguito, per illustrare il concetto vedremo un esempio semplice e "classico", ovvero il problema di cutting stock unidimensionale (anche noto come *problema di sfridi*).

Tale problema può esprimersi come segue. Si devono ricavare, da assi di legno di lunghezza standard pari a L, determinate quantità di tavolette, di m lunghezze diverse. Sia d_i il numero di tavolette di lunghezza l_i che devono essere prodotte $i = 1, \ldots, m$. Ciascun asse può essere tagliato in diversi modi, detti *schemi di taglio*. Ad esempio, se L è pari a 1 metro, è possibile ricavare da un asse: (i) 4 tavolette da 25 cm, oppure (ii) 1 tavoletta da 40 cm e 2 tavolette da 30 cm, o ancora (iii) 1 da 40 cm, 1 da 30 cm e 1 da 25 cm (con uno scarto di 5 cm), etc. Il problema consiste nel determinare il numero minimo di assi da tagliare per soddisfare la domanda di tavolette, e quali schemi di taglio usare per ciascun asse [1]. Sia n il numero complessivo dei possibili schemi di taglio. Se x_j è pari al numero di assi che dovranno essere tagliate secondo lo schema di taglio j-esimo, il problema può essere formulato come segue, ove a_{ij} sta a indicare il numero di tavolette di lunghezza l_i prodotte tagliando un'asse secondo lo schema j-esimo.

$$\begin{aligned} \min\ z = & \sum_{j=1}^{n} x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \geq d_i, i = 1, \ldots, m \\ & x_j \geq 0, j = 1, \ldots, n \end{aligned} \tag{1}$$

*Dipartimento di Ingegneria dell'Informazione - Università di Siena

[1] Ovviamente consideriamo qui solo schemi di taglio "non dominati", cioè tali che l'insieme di tavolette prodotte da uno schema tagliando un asse non è contenuto in nessun insieme prodotto da un altro schema.

Si noti che qui si sta considerando il problema trascurando l'interezza delle x_j, cosa in genere lecita nelle applicazioni reali di questo modello. Qualora ciò non fosse possibile, perché ad esempio i valori d_i sono bassi, la natura del problema cambierebbe sensibilmente e andrebbe affrontato come problema di PLI, con tecniche del tutto diverse.

L'ostacolo che si incontra nell'affrontare il problema (1) in contesti reali sta nel fatto che il numero di schemi di taglio può essere estremamente elevato, tanto che può richiedere troppo tempo anche solo la *generazione* della matrice dei coefficienti del problema (1). Se l'asse è lunga 1 metro, basta che vi siano 6 o 7 tipi di tavoletta per far esplodere il numero di schemi di taglio possibili.

Tuttavia, sappiamo dalla teoria della programmazione lineare che il numero di variabili diverse da zero – e quindi, nel nostro caso, il numero di schemi di taglio effettivamente usati – in una soluzione di base, e dunque anche in quella ottima, non supera m. L'idea allora è quella di trovare il modo di risolvere il problema senza generare *esplicitamente* l'intera matrice A, ma solamente quelle colonne che più delle altre risultano "importanti" per la minimizzazione del numero di assi. Il procedimento che illustreremo per il problema del cutting stock unidimensionale si può applicare a molti altri problemi di PL con un numero elevato di variabili, e rappresenta forse una delle più utili applicazioni della teoria della dualità alla soluzione di problemi reali.

2 Knapsack intero

Prima di occuparci della soluzione del problema del cutting stock, analizziamo un problema combinatorio che ci servirà in seguito. Un'istanza del problema di KNAPSACK INTERO è la seguente, in cui c_j indica l'utilità dell'oggetto j e a_j il suo peso, mentre b è il massimo peso che lo zaino può sopportare.

$$\begin{aligned} \max \ & \sum_{j=1}^{n} c_j x_j & & & (2) \\ & \sum_{j=1}^{n} a_j x_j & \leq & \ b \\ & x_j & \geq & \ 0, j = 1, \ldots, n \\ & x_j & \textit{intero} & \ , j = 1, \ldots, n \end{aligned}$$

Rispetto a KNAPSACK 0-1, in questo caso possiamo prendere un numero qualsiasi (purché intero) di copie di ciascun tipo di oggetto. Al pari di KNAPSACK 0-1, anche KNAPSACK INTERO è risolubile in tempo pseudopolinomiale, con una tecnica molto simile. Sia $F(k, y)$ il valore della soluzione ottima del problema quando si considerino solo

i primi k *tipi di oggetti* ($k \leq n$), e uno zaino di capacità $y \leq b$. Allora è facile verificare che vale la seguente formula di programmazione dinamica:

$$F(k, y) = \max\{F(k-1, y); F(k, y-a_k) + c_k\} \tag{3}$$

Considerando infatti uno zaino di capacità y, e i primi k tipi di oggetti, ci sono solo due possibilità: che il k-esimo tipo di oggetto sia rappresentato nello zaino oppure no. In quest'ultimo caso, chiaramente $F(k, y) = F(k-1, y)$. Altrimenti, nello zaino di capacità y c'è almeno un oggetto di tipo k. Tale oggetto ha un peso pari a a_k. Togliendolo, quindi, si avrebbe come soluzione ottima il valore $F(k, y-a_k)$ (che eventualmente può essere pari a $F(k-1, y-a_k)$). Chiaramente occorre inizializzare il procedimento ponendo $F(k, 0) = 0$ per ogni k, $F(0, y) = 0$ per ogni y e $F(k, y) = -\infty$ per $y < 0$.

Possiamo organizzare i calcoli secondo una matrice $n \times b$, in cui l'elemento $F(n, b)$ indicherà il valore della soluzione ottima del problema. A ogni passo, il valore dell'elemento in posizione (k, y) si ricava confrontando l'elemento superiore con quello che si trova a_k colonne più a sinistra, cui va aggiunto c_k. Ad esempio, se si ha:

$$\begin{aligned} \max\ & 5x_1 + 4x_2 + 6x_3 && && \\ & 3x_1 + 2x_2 + 6x_3 && \leq && 9 \\ & x_j && \geq && 0, \quad j = 1, \ldots, n \\ & x_j && intero && , \quad j = 1, \ldots, n \end{aligned} \tag{4}$$

si ottiene la seguente tabella.

$\downarrow k \rightarrow y$	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	5	5	5	10	10	10	15
2	0	0	4	5	8	9	12	13	16	17
3	0	0	4	5	8	9	12	13	16	**17**

Da questa tabella è facile ricostruire che la soluzione ottima è $x_1 = 1, x_2 = 3, x_3 = 0$. Poiché ciascun elemento di questa matrice può essere calcolato in tempo costante, la complessità di tale algoritmo è $O(nb)$, al pari di quanto accade per KNAPSACK 0-1.

3 Generazione di colonne

Tornando al problema (1), consideriamo il problema duale:

$$\max\ w = \sum_{i=1}^{m} d_i u_i \tag{5}$$

$$\begin{aligned} \sum_{i=1}^{m} a_{ij} u_i &\leq 1, j = 1, \ldots, n \\ u_i &\geq 0, i = 1, \ldots, m \end{aligned} \tag{6}$$

Dunque, se il problema primale ha un numero molto elevato di variabili, il problema duale ha un numero molto elevato di vincoli. L'idea del procedimento risolutivo è allora la seguente. Supponiamo di generare, nel problema primale (1), solo una sottomatrice R di A, con un numero n_R molto ridotto di colonne (ad esempio, generate in modo tale che ciascuna tavoletta sia rappresentata in almeno uno schema di taglio, per garantire che una soluzione ammissibile esista). Chiameremo *primale ristretto* tale problema. Il numero di vincoli del problema primale rimane evidentemente inalterato, in quanto è come se avessimo vincolato a 0 (ossia, eliminato) un gran numero di variabili. Consideriamo adesso il duale del problema ristretto: la sua struttura è del tutto simile a (5), con soli n_R vincoli anziché n, ma con lo stesso insieme di variabili duali. Tale problema è il *duale ristretto*. Risolvendo il problema primale ristretto, si ottiene una soluzione ottima x_R^*, di valore z_R^*, mentre il duale ristretto ha soluzione ottima u_R^*, di valore $w_R^* = z_R^*$. Ora, si possono verificare solo due casi: (i) la soluzione u_R^* è ammissibile per il problema duale originario (5), oppure (ii) no. Nel primo caso, si ha che, pur contenendo il duale solo un numero limitato di vincoli, la soluzione ottenuta soddisfa *tutti* i vincoli (6), e dunque $w_R^* = w^* = z^*$: la soluzione del primale ristretto è dunque ottima per il problema originario (1). Nel secondo caso, si ha invece che almeno un vincolo del problema duale (5) risulta violato da u_R^*, e quindi u_R^* non è ottima per il duale. Identificato un vincolo violato, allora, possiamo pensare di inserirlo nel duale ristretto, ovvero di *generare la variabile* x_j corrispondente a tale vincolo violato e aggiungerla al primale ristretto. Dopo di che, si risolve nuovamente il primale ristretto, ottenendo così una nuova coppia di soluzioni ottime per i problemi (1) e (5), e si itera l'argomento. Il procedimento si arresta quando la soluzione ottima del duale ristretto non viola più nessun vincolo di (5), e dunque la soluzione ottima del primale ristretto è ottima per il problema primale originario.

Il punto cruciale di tutto il procedimento sta nella possibilità di riuscire a certificare, in modo efficiente, che la soluzione ottima del duale ristretto corrente non viola nessuno dei vincoli che non sono stati esplicitamente generati, oppure nel riuscire a trovare almeno uno di tali vincoli violati. Tale problema si chiama *problema di separazione*, e dal modo in cui esso viene risolto dipende sostanzialmente l'efficienza del metodo. Nel seguito, vediamo come tale problema possa essere risolto nel caso del problema di cutting stock unidimensionale.

Supponiamo dunque di avere la soluzione ottima del duale ristretto, in cui indichiamo con u_i^* la generica variabile. Il problema consiste nel trovare un vincolo del tipo (6) violato, o mostrare che non ne esistono. Consideriamo allora un generico schema di taglio, e sia q_i il numero di tavolette di tipo i che tale schema produce. Per tale schema di taglio (non

generato fino ad ora) il termine di sinistra del corrispondente vincolo (6) vale:

$$\sum_{i=1}^{m} q_i u_i^* \tag{7}$$

dove, si noti, le q_i sono le incognite, in quanto definiscono lo schema di taglio che vogliamo trovare. Perché il corrispondente vincolo (6) sia violato, l'espressione (7) deve essere *strettamente* maggiore di 1. Il problema di separazione può dunque essere formulato come un problema di ottimizzazione, in cui cerchiamo lo schema di taglio ammissibile che massimizzi la (7): se, all'ottimo di tale problema, la funzione obiettivo vale più di 1, la soluzione ottima ci consente di identificare un vincolo violato; altrimenti avremo certificato l'ottimalità della soluzione ottima del problema duale ristretto per l'intero problema duale (5)-(6) e, dunque, quella del primale ristretto per l'intero problema originario.

Dunque, il problema di separazione consiste nel trovare i valori q_i in modo che descrivano uno schema di taglio ammissibile e che sia massimizzato (7). Chiaramente, uno schema di taglio è ammissibile se la somma delle lunghezze delle tavolette da esso prodotte non eccede la lunghezza L dell'asse di legno, e dunque in definitiva il problema di separazione è:

$$\begin{aligned} \max & \sum_{i=1}^{m} u_i^* q_i \\ & \sum_{i=1}^{m} l_i q_i \quad \leq \quad L \\ & q_i \quad \geq \quad 0, \quad i = 1, \ldots, m \\ & q_i \quad intero \quad , \quad i = 1, \ldots, m \end{aligned} \tag{8}$$

evidentemente il problema di separazione è un'istanza di KNAPSACK INTERO, in cui i tipi di tavolette giocano il ruolo degli oggetti da inserire nello zaino, e la lunghezza dell'asse quello della capacità dello zaino. L'estrema efficienza con cui è possibile, in pratica, risolvere problemi di knapsack intero consente di risolvere problemi di cutting stock unidimensionali. Ricapitolando, lo schema risolutivo è dunque il seguente.

Algoritmo COLUMN GENERATION;

1. Si genera un insieme iniziale R di colonne della matrice A (schemi di taglio), tali che il problema primale ristretto abbia soluzione ammissibile;

2. Si risolve il primale con insieme di colonne dato da R, sia x_R^* la soluzione ottima di tale problema e u_R^* la soluzione ottima del suo duale;

3. si risolve il problema di separazione (8) definito da u_R^*, sia q^* la sua soluzione ottima;

4. Se $(u_R^*)^T q^* \leq 1$, stop (x_R^* è la soluzione ottima), altrimenti aggiungi a R la colonna data dal vettore q^*, e vai al passo 2.

Essendo finito (benché eventualmente molto alto) il numero dei possibili schemi di taglio, la convergenza del metodo è garantita. C'è da aggiungere che nonostante, in linea teorica, nulla garantisca che il numero di problemi di separazione da risolvere sia piccolo, in pratica esso è estremamente basso rispetto al totale del numero degli schemi di taglio possibili. Intuitivamente, ciò è dovuto al fatto che il procedimento genera solo quelle colonne che sono effettivamente importanti ai fini della minimizzazione della funzione obiettivo del problema, che in definitiva non saranno mai un numero enorme. Il fatto che in taluni casi la convergenza del metodo sia lenta, può dipendere dall'esistenza di molte soluzioni di base degeneri. Può infatti verificarsi che l'aggiunta di una colonna all'insieme R non determini alcun miglioramento nel valore ottimo del problema primale ristretto. Tuttavia, raramente questo fatto impedisce l'applicazione del metodo.