Metodi e Modelli per l'Ottimizzazione Combinatoria

Cover inequalities

L. De Giovanni M. Di Summa

In questa lezione introdurremo una classe di disuguaglianze, dette *cover inequalities*, che permettono di migliorare la formulazione del problema dello zaino. Vedremo poi come tali disuguaglianze trovino applicazione in problemi di programmazione lineare intera più generali.

1 Cover inequalities per il problema dello zaino

Nel problema dello zaino (detto anche *knapsack*) sono dati n oggetti di peso $a_1, \dots, a_n$ e profitto $p_1, \dots, p_n$ ciascuno, ed un contenitore (lo zaino) la cui portata massima è β. Si chiede di determinare un sottoinsieme degli n oggetti che possa essere sistemato nel contenitore senza eccedere la sua portata massima e che dia il massimo profitto possibile. Nel seguito assumeremo che $a_1, \dots, a_n$ e β siano tutti numeri interi.[1] Usando variabili binarie $x_1, \dots, x_n$, dove $x_i = 1$ se e solo se l'oggetto i viene selezionato, possiamo formulare il problema dello zaino tramite il seguente programma lineare intero:

$$\max \sum_{i=1}^{n} p_i x_i \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{n} a_i x_i \leq \beta, \tag{2}$$

$$0 \leq x_i \leq 1, \quad i = 1, \dots, n, \tag{3}$$

$$x_i \in \mathbb{Z}, \quad i = 1, \dots, n. \tag{4}$$

La regione ammissibile del rilassamento continuo, data dai vincoli (2)–(3), è generalmente molto lontana dalla formulazione ideale del problema. Si può dunque pensare di introdurre delle disuguaglianze che migliorino la formulazione: tali disuguaglianze devono essere soddisfatte da tutte le soluzioni intere del problema, ma devono dar luogo, se possibile, ad un rilassamento continuo più vicino alla formulazione ideale.

[1] Ricordiamo che dal punto di vista implementativo lavoriamo sempre con numeri razionali. Partendo da valori razionali di $a_1, \dots, a_n$ e β, possiamo ridurci a valori interi moltiplicando questi valori per il loro minimo comune denominatore. Il problema risultante è del tutto equivalente a quello dato, in quanto stiamo solo cambiando l'unità di misura del peso.

1.1 Cover inequalities

Chiamiamo *cover* un qualunque sottoinsieme degli n oggetti che ecceda la portata dello zaino: dunque una cover è un sottoinsieme $C \subseteq \{1, \ldots, n\}$ tale che

$$\sum_{i \in C} a_i > \beta.$$

Ricordando che $a_1, \ldots, a_n$ e β sono tutti interi, la condizione può essere riscritta così:

$$\sum_{i \in C} a_i \geq \beta + 1.$$

Data una cover C, poiché gli oggetti di C non possono essere sistemati nello zaino tutti assieme, ne segue che almeno uno di tali oggetti dovrà essere sacrificato. In altre parole, dei $|C|$ oggetti di C, al massimo $|C|-1$ possono essere selezionati. Vale dunque la seguente disuguaglianza:

$$\sum_{i \in C} x_i \leq |C| - 1.$$

Tale disuguaglianza è detta *cover inequality*.

Aggiungendo tutte le cover inequalities alla formulazione originale del problema dello zaino, si ottiene una formulazione migliore (si può dimostrare, tuttavia, che quella ottenuta non è ancora la formulazione ideale). Purtroppo, però, il numero di cover inequalities è esponenzialmente grande, in quanto nel caso peggiore ne abbiamo una per ogni sottoinsieme non vuoto di $\{1, \ldots, n\}$. Anche se, nei casi pratici, solo una parte dei sottoinsiemi di $\{1, \ldots, n\}$ sono cover, il numero di cover è comunque troppo grande per poter pensare di aggiungere alla formulazione tutte le cover inequalities. Ci poniamo dunque l'obiettivo di aggiungere tali disuguaglianze dinamicamente solo quando sono realmente necessarie.

1.2 Separazione delle cover inequalities

Supponiamo di aver risolto il rilassamento continuo (1)–(3) (ad esempio col metodo del simplesso), ottenendo una soluzione $\bar{x}$ non intera. Ci chiediamo se esista una cover inequality violata da $\bar{x}$, con l'obiettivo di aggiungere tale disuguaglianza alla formulazione e risolvere nuovamente il programma lineare ottenuto. Il problema di trovare una cover inequality violata da $\bar{x}$, o decidere che non ce ne sono, è il *problema di separazione* per le cover inequalies.

Decidere se esiste una cover inequality violata da $\bar{x}$ significa decidere se esiste un sottoinsieme $C \subseteq \{1, \ldots, n\}$ tale che

(i) $\sum_{i \in C} a_i \geq \beta + 1$;

(ii) $\sum_{i=1}^{n} \bar{x}_i > |C| - 1$.

La condizione (i) definisce una cover, mentre la (ii) dice che $\bar{x}$ non soddisfa la cover inequality corrispondente a C.

Introduciamo le variabili binarie $z_1, \ldots, z_n$, dove $z_i = 1$ se e solo se l'oggetto i fa parte della cover C. La condizione (i) può essere scritta come segue:

$$\sum_{i=1}^{n} a_i z_i \geq \beta + 1.$$

Notando che $\sum_{i=1}^{n} z_i = |C|$, possiamo riscrivere la condizione (ii) così:

$$\sum_{i=1}^{n} \bar{x}_i z_i > \sum_{i=1}^{n} z_i - 1, \qquad \text{cioè} \qquad \sum_{i=1}^{n} (1 - \bar{x}_i) z_i < 1.$$

Consideriamo il seguente problema di programmazione lineare intera:

$$\min \ \sum_{i=1}^{n} (1 - \bar{x}_i) z_i \tag{5}$$

$$\text{s.t.} \ \sum_{i=1}^{n} a_i z_i \geq \beta + 1, \tag{6}$$

$$0 \leq z_i \leq 1, \quad i = 1, \ldots, n, \tag{7}$$

$$z_i \in \mathbb{Z}, \quad i = 1, \ldots, n. \tag{8}$$

I vincoli del problema assicurano che le z_i definiscano una cover. Sia $\bar{z}$ la soluzione ottima (intera) di questo problema. Se il valore ottimo è inferiore a 1, allora $\sum_{i=1}^{n} (1 - \bar{x}_i)\bar{z}_i < 1$, il che, come visto, significa che la cover inequality definita da $\bar{z}$ è violata da $\bar{x}$. Altrimenti, se il valore ottimo è ≥ 1, allora non esiste alcuna cover inequality violata, in quanto in questo caso $\sum_{i=1}^{n} (1 - \bar{x}_i) z_i \geq 1$ per qualunque vettore z che definisce una cover.

Possiamo quindi decidere se esiste una cover inequality violata da $\bar{x}$ (e in tal caso trovarla) risolvendo il problema di programmazione lineare intera (5)–(8). Si noti che tale problema è molto simile al problema dello zaino (1)–(4) da cui siamo partiti, dunque apparirebbe molto più sensato risolvere direttamente il problema iniziale piuttosto che risolverne uno molto simile solo per trovare una disuguaglianza da aggiungere alla formulazione. Vedremo tra un attimo, tuttavia, come questo approccio sia molto più promettente in situazioni più generali.

2 Cover inequalities per problemi binari generici

Consideriamo un generico problema di programmazione lineare intera in cui tutte le variabili sono binarie:

$$\max \ c^T x \tag{9}$$

$$\text{s.t.} \ Ax \leq b, \tag{10}$$

$$0 \leq x_i \leq 1, \quad i = 1, \ldots, n, \tag{11}$$

$$x_i \in \mathbb{Z}, \quad i = 1, \ldots, n. \tag{12}$$

L'osservazione cruciale è che ogni singolo vincolo del sistema $Ax \leq b$ può essere visto come un vincolo di tipo knapsack (2). In altre parole, se sostituiamo il sistema $Ax \leq b$ con uno qualunque dei suoi vincoli, eliminando tutti gli altri, otteniamo un rilassamento del problema che ha esattamente la forma del problema dello zaino. È dunque possibile aggiungere alla formualzione le cover inequalities associate a ciascuno dei problemi dello zaino che si ottengono in questo modo (eliminando cioè tutti i vincoli tranne uno). Anche in questo caso il numero di cover inequalities sarà enorme, dunque ci proponiamo di usare quanto visto sopra per aggiungere tali disuguglianze all'occorrenza.

Supponiamo di aver risolto il rilassamento continuo (9)–(11), ottenendo una soluzione non intera $\bar{x}$. Per ogni problema dello zaino ottenuto rimuovendo tutti i vincoli del sistema $Ax \leq b$ tranne uno, ci chiediamo se esista una cover inequality violata da $\bar{x}$. A questo scopo, è sufficiente risolvere un problema della forma (5)–(8). Se il valore ottimo di tale problema è inferiore a 1, allora otteniamo una cover inqueality violata da $\bar{x}$ che possiamo aggiungere alla formulazione; altrimenti non ci sono cover inequalities violate da $\bar{x}$. A questo punto possiamo risolvere nuovamente il rilassamento continuo comprensivo delle disuguaglianze aggiunte ed iterare il procedimento fino a quando si scopre che la soluzione corrente soddisfa tutte le cover inequalities. L'algoritmo proposto è schematizzato qui sotto:

Generazione di cover inequalities

1. Si risolva il rilassamento continuo (9)-(11) e sia $\bar{x}$ la soluzione ottima trovata.
2. Se $\bar{x}$ è intera allora STOP: soluzione ottima intera.
3. Per ogni vincolo del sistema $Ax \leq b$, si risolva il corrispondente programma lineare intero (5)-(8) (dove $a_1, \ldots, a_n$ e β sono rispettivamente i coefficienti ed il termine noto del vincolo considerato). Sia $\bar{z}$ la soluzione ottima intera trovata.
4. Se il valore ottimo del problema (5)-(8) è inferiore a 1, allora $\bar{z}$ individua una cover inequality violata, definita da $C = \{i : \bar{z}_i = 1\}$.
5. Se per tutti problemi (5)-(8) risolti al passo precedente il valore ottimo è ≥ 1, allora non ci sono cover inequalities violate: STOP.
6. Si aggiungano alla formualzione le cover inequalities violate che sono state trovate, si risolva il programma lineare ottenuto, si chiami $\bar{x}$ la soluzione ottima trovata e si torni al passo 2.

Si noti che l'algoritmo prevede di risolvere numerosi problemi della forma (5)-(8), che sono problemi di programmazione lineare intera e dunque in generale particolarmente difficili. Tuttavia, il problema (5)-(8) è uno dei più semplici problemmi di programmazione

lineare intera e dunque, sebbene in linea di principio potrebbe essere necessario un tempo esponenziale per risolverlo, nella pratica si giunge alla soluzione ottima in tempi molto ragionevoli.

Come detto, l'algoritmo termina quando la soluzione corrente non viola alcuna delle cover inequalities associate ai vincoli del problema. È tuttavia possibile interrompere l'algoritmo prematuramente, qualora si ritenga di aver aggiunto un numero di disuguaglianze sufficiente a definire una buona formulazione del problema. In entrambi i casi, se la soluzione $\bar{x}$ non è ancora intera, si può applicare il branch-and-bound. Il fatto di aver aggiunto le cover inequalities permette di partire da una formulazione migliore e dà ragionevoli speranze che il branch-and-bound termini più rapidamente. Questo tipo di approccio, in cui si rafforza la formulazione del rilassamento continuo e poi si applica il branch-and-bound, è detto *cut-and-branch*. Se si decide di aggiungere disuguaglianze per rafforzare la formulazione in tutti i nodi dell'albero di branch-and-bound, si parla di *branch-and-cut*.