Problemi di Partizione e Copertura

Federico Poloni

14 luglio 2004

1 Descrizione del problema

Abbiamo una "scatola" vuota formata da n caselle di forma quadrata e un certo insieme J di pezzi, ognuno dei quali ricopre esattamente un sottoinsieme delle caselle. A ogni pezzo j è associato un "costo" c_j positivo. Possiamo vedere un semplice esempio di questa situazione in figura 1.

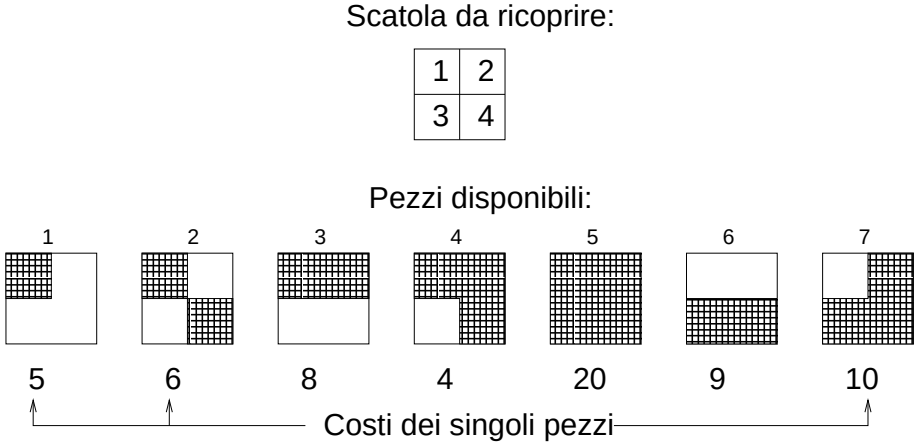

Figura 1: Un semplice esempio di problema CP/PP

Siamo interessati a risolvere i seguenti problemi:

Problema 1 (Covering Problem,CP) *Trovare la copertura di costo totale minimo, cioè il sottoinsieme J' di J tale che:*

1. *Ognuna delle caselle è ricoperta da* almeno *un pezzo contenuto in J';*
2. *La somma dei "costi" dei pezzi di J' è minima tra tutti i sottoinsiemi che soddisfano (1).*

Problema 2 (Partitioning Problem,PP) *Trovare la partizione di costo totale minimo, cioè il sottoinsieme J'' di J tale che:*

1. *Ognuna delle caselle è ricoperta da* esattamente *un pezzo contenuto in J'';*
2. *La somma dei "costi" dei pezzi di J'' è minima tra tutti i sottoinsiemi che soddisfano (1).*

Nel caso della figura, ad esempio, si vede che il PP ha soluzione $J'' = \{1, 7\}$, con costo totale 15, mentre il CP ha soluzione $\{4, 6\}$, con costo totale 13.

2 Esempi del problema CP/PP nelle applicazioni

I problemi reali che possono essere modellizzati come CP sono diversi; ad esempio:

ricerca di informazioni su reti di computer: Su una rete di calcolatori, una grande quantità di dati può essere divisa su più *server*. Ogni server contiene un diverso sottoinsieme dei dati, e impiega un tempo diverso (che possiamo pensare come il costo c_j) per fornire i dati di cui dispone. Quali server dobbiamo interrogare per ottenere tutti i dati nel tempo minore? [1]

Il fattorino della pizzeria: Una pizzeria deve consegnare a domicilio un certo numero di pizze. Pensiamo alle pizze da consegnare come alle caselle da ricoprire; lungo ogni possibile percorso che il fattorino può prendere, riesce a consegnare alcune delle pizze (quelle i cui destinatari abitano lungo la strada percorsa); il costo c_j di ogni percorso può essere il consumo di benzina oppure una funzione più complicata che tiene conto del tempo di consegna.

Sconnettere di un grafo: Le caselle siano un insieme I di percorsi su un grafo, e i possibili pezzi i lati del grafo. Diciamo che il pezzo j ricopre la casella i se il lato j è contenuto nel percorso i. Se c_j è il costo per "rimuovere" il lato j dal grafo, una copertura ottimale corrisponde all'insieme "più economico" di lati da rimuovere per disconnettere tutti i percorsi I.

Colorazione di mappe: Vogliamo colorare le regioni di una mappa con il minimo numero possibile di colori, in modo che due regioni adiacenti abbiano colori diversi. Le regioni qui sono le caselle da ricoprire, e i pezzi sono tutti i possibili insiemi di caselle non adiacenti a due a due. Tutti i costi sono unitari. Se la soluzione del PP contiene n pezzi, allora il minimo numero di colori necessario per colorare la mappa è n.

Problemi sui polimini: Consideriamo ad esempio un problema classico: i dodici *pentamini* (vedi figura 2) possono essere descritti come le figure connesse ottenibili incollando cinque quadrati uguali. I pentamini hanno area totale $12 \cdot 5 = 60$. È possibile usando una e una sola copia di ogni pentamino ricoprire esattamente rettangoli di dimensioni 3×20, 4×15, 5×12, 6×10? Il problema è un semplice problema di partizione (con tutti i costi unitari); l'unica complicazione è che qui i pezzi non sono semplicemente i pentamini, ma ogni possibile posizionamento del pentamino all'interno del rettangolo costituisce un pezzo. Ad esempio, il pezzo a

[1] il DNS, cioè il servizio che su internet traduce gli indirizzi (ad esempio `www.dm.unipi.it`) nella loro rappresentazione "interna" numerica (ad esempio `131.114.72.19` per l'indirizzo qui sopra) è un esempio del caso in esame. Si noti che giocando opportunamente con la funzione costo è possibile minimizzare anche il tempo massimo di accesso: ad esempio, se il tempo di accesso del server è n_j e la somma dei tempi di accesso è M, si dimostra che ponendo $c_j = M^{n_j}$ otteniamo un CP che minimizza il tempo di accesso massimo.

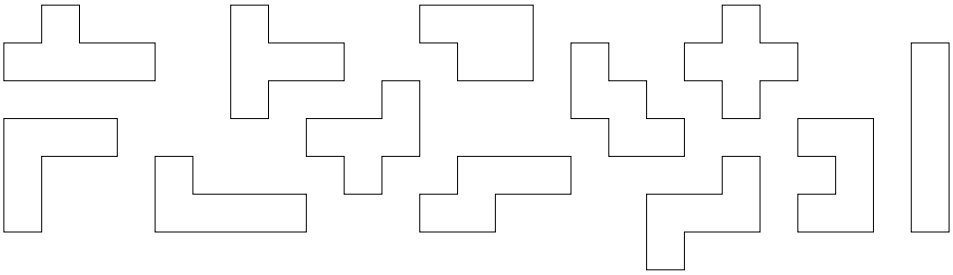

Figura 2: I 12 pentamini

croce può essere messo in 18 modi diversi all'interno del rettangolo 3×20, quindi ai nostri fini costituisce 18 pezzi distinti. In questo modo quindi il numero dei pezzi (e quindi la dimensione del problema) diventa molto grande; esistono in realtà strategie più efficienti della riduzione al PP per risolvere questo problema.

3 Formalizzazione del problema

Sia $I = \{1 \dots m\}$ l'insieme delle caselle da ricoprire, e $J = \{1, \dots n\}$ l'insieme dei pezzi, dove possiamo identificare un pezzo $j \in J$ con il sottoinsieme $P_j \subseteq I$ delle caselle da lui ricoperte. Associamo al problema la matrice $A \in R^{m \times n}$ definita da

$$a_{ij} = \begin{cases} 1 & \text{se il pezzo } j \text{ ricopre la casella } i, \text{ cioè se } i \in P_j \\ 0 & \text{altrimenti} \end{cases}$$

e il vettore dei costi $c \in \mathbb{R}^n_+$. Una soluzione del problema può essere descritta come il vettore $x \in \mathbb{R}^n$, dove

$$x_j = \begin{cases} 1 & \text{se il pezzo } j \text{ viene usato nella copertura} \\ 0 & \text{altrimenti} \end{cases}$$

In questo modo, la condizione che la casella i-esima sia coperta almeno una volta si scrive

$$\sum_{j=0}^{n} a_{ij} x_j \geq 1$$

e quindi la formulazione del CP è

$$\begin{cases} \min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \geq 1 \text{ per } i = 1, 2, \dots m \\ x_j \in \{0, 1\} \text{ per } j = 1, 2, \dots n \end{cases} \tag{1}$$

o in forma matriciale

$$\begin{cases} \min \langle c, x \rangle \\ Ax \geq e \\ x \in \{0, 1\}^n \end{cases} \tag{2}$$

(dove $e = (1, 1, \dots 1) \in \mathbb{R}^n$)

Il PP si formula in modo analogo, solamente abbiamo delle uguaglianze al posto delle disuguaglianze:

$$\begin{cases} \min \sum_{j=1}^{n} c_j x_j \\ \sum_{j=1}^{n} a_{ij} x_j \geq 1 \text{ per } i = 1, 2, \dots m \\ x_j \in \{0, 1\} \text{ per } j = 1, 2, \dots n \end{cases} \tag{3}$$

o in forma matriciale

$$\begin{cases} \min \langle c, x \rangle \\ Ax = e \\ x \in \{0, 1\}^n \end{cases} \tag{4}$$

Nell'esempio di figura 1, ad esempio, numerando pezzi e caselle come nella figura, la matrice A è

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

mentre il corrispondente vettore c dei costi è

$$\begin{pmatrix} 5 & 6 & 8 & 4 & 20 & 9 & 10 \end{pmatrix}$$

4 Riduzione del PP al CP

Proposizione 3 *È possibile attraverso una trasformazione del vettore c dei costi ricondurre il PP a un caso particolare del CP.*

Agiamo in questo modo: sia M un intero positivo abbastanza grande da aversi $M > \sum_{j=0}^{n} c_j$; modifichiamo il vettore c in questo modo: se il pezzo j ha costo c_j e ricopre r_j caselle dell'insieme I, il suo nuovo costo diventa

$$C_j = c_j + r_j M$$

Ora, cerchiamo una soluzione del CP che ha gli stessi dati del PP di partenza, ma ha C come vettore dei costi. Essa sarà la copertura J' che minimizza

$$\sum_{j \in J'} C_j = \sum_{j \in J'} c_j + M \sum_{j \in J'} r_j$$

Poiché M è molto grande rispetto ai c_j, la soluzione al CP è la copertura J' che soddisfa questi due vincoli:

- J' minimizza $\sum_{j \in J'} r_j$, cioè i pezzi di J' ricoprono (in totale) il minor numero possibile di caselle
- Fra i $\bar{J}$ che soddisfano il punto precedente, J' è quello con il costo minore

Ora, se esiste almeno una partizione costruibile con i pezzi di J, le partizioni sono tutte e sole le coperture che soddisfano la prima condizione (perché ogni casella è ricoperta almeno una volta per ogni copertura, ed *esattamente* una volta nel caso delle partizioni); quindi la soluzione del CP modificato è la soluzione

del PP. D'altro canto, se non esiste alcuna partizione costruibile con i pezzi di J, allora la soluzione del CP modificato sarà maggiore di $(n+1)M$ (perché ogni copertura ricopre almeno un pezzo più di una volta), quindi possiamo ricavare dalla soluzione del CP l'informazione che il PP non ammette soluzione. □

Nelle sezioni successive possiamo quindi ridurci a considerare solo il CP ed enunciare l'algoritmo risolutivo unicamente per questo problema.

5 Prime semplificazioni

Sono possibili *a priori* alcune semplificazioni volte a ridurre le dimensioni della matrice di lavoro A e quindi consentire una soluzione più agevole. Elenchiamo le principali; siano qui di seguito r_i l'i-esima riga e a_j la j-esima colonna di A:

1. Se una riga r_i contiene unicamente zeri, allora il problema è insolubile: infatti l'i-esima equazione diventa $0 \geq 1$.

2. Se una colonna a_i contiene unicamente zeri, essa può essere eliminata: infatti il pezzo corrispondente non ricopre alcuna casella ed è perciò inutile.

3. Se per una riga r_i si ha $r_i = e_k$ (dove e_k è il k-esimo vettore della base canonica di $\mathbb{R}^m$), allora dev'essere $x_k = 1$ in ogni soluzione accettabile (perché la i-esima equazione diventa $x_k \geq 1$). Possiamo quindi eliminare la k-esima colonna e tutte le righe r_i tali che $a_{ik} = 1$ (infatti, se $x_k = 1$ le equazioni corrispondenti sono soddisfatte in ogni caso).

4. Se $r_i \geq r_k$ per due righe qualunque r_i e r_k, allora r_i può essere eliminata: infatti l'equazione corrispondente è automaticamente soddisfatta da tutte le soluzioni che soddisfino anche la k-esima equazione. In termini di caselle e pezzi, ogni pezzo che ricopre la casella k ricopre anche la casella i, quindi imporre che i sia ricoperta è superfluo.

5. Se $a_l \geq a_j$ per due colonne qualunque a_l e a_j, e inoltre $c_l \leq c_j$, allora a_j può essere eliminata: infatti ai fini della copertura a_j è inutile, in quanto a_l ricopre le stesse caselle con un costo minore.

6. Se per un certo insieme L di colonne e per una colonna a_k si ha $\sum_{j \in L} a_l \geq a_k$ e $\sum_{j \in L} c_j \leq c_k$, allora a_k può essere eliminata. Infatti, come nel caso precedente, l'insieme di colonne L ricopre con un costo minore le stesse caselle che sono ricoperte anche da a_k.

Si noti che l'ultima riduzione è particolarmente onerosa in termini computazionali, quindi la convenienza o meno (in termini di tempo di elaborazione) dell'effettuarla dipende dal problema specifico. Inoltre, alcune di queste semplificazioni possono essere opportunamente modificate per semplificare ulteriormente il PP *prima* della modifica dei costi come indicato nella sezione 4

A titolo di esempio applichiamo queste riduzioni al problema della figura 1:

1. Le colonne 1, 2 e 3 sono dominate dallla colonna 4, possono essere eliminate (riduzione 5).

2. Le colonne 4 e 6 dominano la colonna 5, quindi quest'ultima può essere eliminata (riduzione 6).

3. La riga 1 (dopo le riduzioni precedenti) contiene un solo elemento uguale a 1, nella colonna 4, quindi dev'essere $x_4 = 1$, e di conseguenza possiamo eliminare le righe 1, 2 e 4. (riduzione 3).

4. A questo punto, la colonna 7 è dominata dalla colonna 6 e quindi può essere eliminata (riduzione 5).

   A questo punto il problema è banale. Quindi la semplice applicazione di queste riduzioni ci ha permesso di risolvere completamente il problema senza la necessità di passare dall'algoritmo risolutivo.

6 Richiami di programmazione lineare

Modifichiamo leggermente il CP per porlo nella forma canonica dei problemi di programmazione lineare (LP): sia $s \in \mathbb{N}_+^m$ il "vettore degli scarti" definito come $s = Ax - b$ (s ha elementi interi e positivi per i vincoli del CP originale). Allora possiamo pensare alle s_i come a nuove incognite del problema e riformularlo così:

$$\begin{cases} \min \langle c, x \rangle \\ Ax - s = e \\ x \in \{0,1\}^n \\ s \in \mathbb{N}^m \end{cases} \tag{5}$$

o anche, estendendo lo spazio di definizione del problema e ponendo:

$$\begin{aligned} \bar{x} = &\quad (x \mid s) &&\in \mathbb{R}^{n+m} \\ \bar{c} = &\quad (c \mid 0) &&\in \mathbb{R}^{n+m} \\ \bar{A} = &\quad [A \mid -I] &&\in \mathbb{R}^{m \times (n+m)} \end{aligned}$$

riformuliamo il CP come

$$\begin{cases} \min \langle \bar{c}, \bar{x} \rangle \\ \bar{A}\bar{x} = e \\ x \in \{0,1\}^n \\ s \in \mathbb{N}^m \end{cases} \tag{6}$$

Possiamo ora applicare una manipolazione algebrica comune nei problemi di LP: innanzitutto, definiamo

Definizione 4 *Una* soluzione di base *è una soluzione ammissibile per il sistema $Ax = b$ (con $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$) è una soluzione tale che sia possibile suddividere la matrice A (isolando alcune colonne non necessariamente nella posizione iniziale) come $[B \mid N]$, e x nello stesso modo come $(x_B \mid x_N)$, in modo che:*

1. *B sia una matrice quadrata $m \times m$ invertibile*
2. *$x_N = 0$*

Si noti che in questo modo

$$b = Ax = [B \mid N](x_B \mid 0) = Bx \Longrightarrow x = B^{-1}b$$

Ora, con una semplice manipolazione algebrica si prova il seguente risultato:

Proposizione 5 *Sia $x^* = (B^{-1}b \mid 0)$ una soluzione di base del sistema*

$$[B \mid N](x_B \mid x_N) = b \tag{7}$$

$c = (c_B \mid c_N) \in \mathbb{R}^n$ e $x = (x_B, x_N)$ un'altra soluzione di (7). Allora condizione necessaria e sufficiente perché $\langle c, x^* \rangle < \langle c, x \rangle$ è che

$$\langle c_N - c_B B^{-1} N, x_N \rangle < 0 \tag{8}$$

Dimostrazione: poiché x risolve il sistema,

$$Bx_B + Nx_N = b \Longrightarrow x_B = B^{-1}b - B^{-1}Nx_N$$

Quindi sostituiamo nel prodotto scalare per ottenere

$$\begin{aligned} \langle c, x \rangle &= \langle c_B, x_B \rangle + \langle c_N, x_N \rangle = \\ &= \langle c_B, B^{-1}b - B^{-1}Nx_N \rangle + \langle c_N, x_N \rangle = \\ &= \langle c_B, B^{-1}b \rangle - \langle c_B, B^{-1}Nx_N \rangle + \langle c_N, x_N \rangle = \\ &= \langle c_B, B^{-1}b \rangle + \langle c_N - c_B B^{-1} N, x_N \rangle \end{aligned}$$

Da cui è immediata la tesi. □

La quantità $c_N - c_B B^{-1} N$ è detta *gradiente ridotto* del problema di minimo.

7 Coperture prime e soluzioni di base

Definizione 6 *Una copertura $J' \subseteq J$ si dice* prima *se ogni suo sottoinsieme proprio $J'' \subsetneq J'$ non è una copertura.*

ossia, intuitivamente, una copertura è prima se non contiene "pezzi superflui". È evidente il seguente fatto:

Proposizione 7 *La soluzione del CP è una copertura prima.*

Proveremo in questa sezione che

Proposizione 8 *Ogni copertura prima J' per il CP*

$$\begin{cases} \min \langle \bar{c}, \bar{x} \rangle \\ \bar{A}\bar{x} = e \\ x \in \{0,1\}^n \\ s \in \mathbb{N}^m \end{cases}$$

si può estendere a una soluzione di base del sistema $\bar{A}\bar{x} = e$

Dimostrazione: Costruiremo la matrice di base B relativa a questa soluzione scegliendo le colonne opportune di $\bar{A}$. Sia r il numero di pezzi contenuti in J'. Notiamo che il fatto che J' sia prima è equivalente ad affermare che per ogni pezzo $j \in J'$ esiste almeno una riga i di A tale che $a_{ij} = 1$ e $a_{ik} = 0$ per $k \neq j$. Infatti, se così non fosse $J' \setminus \{j\}$ resterebbe ancora una copertura. (in termini di matrici, ciò significa che le r colonne $j \in J'$ della matrice A contengono al loro interno una matrice unitaria $r \times r$).

Pertanto, scegliamo per ogni colonna j una riga i_j che soddisfi questa condizione (in generale ce ne può essere anche più di una) e definiamo la base B come formata dalle r colonne di J' in A e dalle colonne (in $-I$) corrispondenti alle variabili di scarto delle $m - r$ righe che *non* abbiamo scelto come i_j per nessun j.

Ad esempio, nel problema di figura 1, possiamo scegliere $J' = \{3, 7\}$. Nella prima riga compare il vettore $(1, 0)$ (perché la casella 1 è coperta dal solo pezzo 3), quindi possiamo scegliere $i_1 = 1$; nella terza e quarta riga compare il vettore $(0, 1)$, quindi possiamo scegliere arbitrariamente $i_2 = 3$ o 4. Scegliamo $i_2 = 3$; le variabili di scarto da aggiungere in base sono allora quelle della 2^a e 4^a riga: quindi,

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$

Torniamo al caso generale; è possibile verificare che la soluzione $x = B^{-1}b$ costruita con questa matrice B corrisponde alla copertura J' di partenza; ora, però, dobbiamo verificare che B sia una matrice invertibile. Per fare questo, applichiamo una permutazione delle righe che porti nelle prime r righe la matrice identità che abbiamo dimostrato esistere nelle prime r colonne, e scriviamo le colonne delle variabili di scarto nell'ordine con cui le abbiamo estratte da $-I$: in questo modo si ottiene una matrice della forma

$$\hat{B} = \left[\begin{array}{c|c} I & 0 \\ \hline * & -I \end{array}\right]$$

(dove gli elementi segnati con $*$ sono valori 0 o 1 qualunque). È evidente che questa matrice è invertibile in quanto triangolare;inoltre si verifica moltiplicando per sottomatrici che $\hat{B}\hat{B} = I$, cioè $\hat{B}$ è *involutoria*[2]. Quindi, poiché $\hat{B} = PB$ (P sia un'opportuna matrice di permutazione, che se moltiplicata a sinistra per B ne permuta le righe) e P e $\hat{B}$ sono invertibili, ne segue che anche B è invertibile. □

Questo completa la nostra dimostrazione che a ogni J' si associa una soluzione di base. Possiamo però fare altre osservazioni che ci torneranno utili quando dovremo calcolare il gradiente ridotto del nostro problema di minimo. Innanzitutto, possiamo scrivere

$$\hat{B}^{-1} = \hat{B} \Longrightarrow (PB)^{-1} = PB \Longrightarrow B^{-1} = PBP = \hat{B}P$$

Poi, notiamo che $c_B\hat{B} = c_B$: infatti, scomponiamo c_B separando le colonne relative a variabili di A e quelle relative a variabili di scarto (che sono tutte

[2]Una matrice M si dice involutoria quando $M^{-1} = M$, cioè M è l'inversa di sé stessa

nulle, perché era $\bar{c} = (c \mid 0)$):

$$c_B \hat{B} = (c'_B \mid 0) \left[\begin{array}{c|c} I & 0 \\ \hline * & -I \end{array}\right] = (c'_B \mid 0)$$

come si ricava moltiplicando per sottomatrici.

In conclusione, essendo $\hat{B} = PB$ e $\hat{N} = PN$, possiamo scrivere il gradiente ridotto nella forma

$$c_N - c_B B^{-1} N = c_N - c_B(\hat{B}P)N = c_N - (c_B \hat{B})(PN) = c_N - c_B \hat{N}$$

che computazionalmente è più semplice perché non richiede di invertire la matrice B, ma semplicemente di applicare una permutazione di righe a N.

8 Piani secanti

Enunceremo ora l'idea su cui si fonda l'algoritmo dei piani secanti. Consideriamo una soluzione di base $x^* = (B^{-1}b \mid 0)$ del CP: sotto quali condizioni è possibile che la funzione obiettivo $\langle c, x \rangle$ raggiunga un valore strettamente minore di quello raggiunto in x^*? La proposizione 5 ci viene in aiuto: è necessario che

$$\langle c_N - c_B \hat{N}, x_N \rangle < 0$$

per la nuova soluzione $x = (x_B \mid x_N)$. Detto g il gradiente ridotto, espandiamo il prodotto scalare e riscriviamo:

$$\sum_{i=1\ldots n-m} g_i \, (x_N)_i < 0$$

Perché questo avvenga, poiché gli elementi di x_N sono tutti positivi o nulli, condizione *necessaria* (ma non sufficiente!) è che per un certo i si abbia contemporaneamente $g_i < 0$ e $(x_N)_i > 0$ (altrimenti la somma sarebbe solo su addendi positivi o nulli...). Allora, abbiamo

Proposizione 9 *Data x^* soluzione di base per il CP, tutte le soluzioni x tali che $\langle c, x \rangle < \langle c, x^* \rangle$ verificano*

$$\begin{gathered} (x_N)_i > 0 \text{ per almeno un } i \text{ tale che } g_i < 0 \\ \Updownarrow \\ \sum_{i:g_i<0} (x_N)_i > 0 \\ \Updownarrow \\ \sum_{i:g_i<0} (x_N)_i \geq 1 \end{gathered} \tag{9}$$

le ultime equivalenze valgono perché gli elementi di x_N sono interi e positivi.

Notiamo inoltre che l'insieme $Q = \{i : g_i < 0\}$ contiene solo variabili del problema originale e non variabili di scarto: infatti, sulle colonne corrispondenti a variabili di scarto g_i è sicuramente non negativo, perché nell'espressione

$$g = c_N - c_B \hat{N}$$

abbiamo: $c_B \geq 0$ per le condizioni del problema; $\hat{N}$ ha elementi negativi o nulli nelle colonne corrispondenti alle variabili di scarto (che, ricordiamo, sono quelle della parte $-I$ della matrice $\bar{A}$), e c_N ha elementi nulli nelle colonne corrispondenti alle variabili di scarto (perché corrispondono agli zeri aggiunti per trasformare c in $\bar{c}$).

Quindi, l'osservazione fondamentale è che *il vincolo qui trovato è della forma* $\sum_{i \in S} x_i \geq 1$ *per un qualche insieme* S*, cioè dello stesso tipo dei vincoli del problema originale (1).* Quindi, se aggiungiamo questa condizione al problema originale otteniamo un nuovo problema di tipo CP.

9 Algoritmo dei piani secanti

L'idea è quella di spezzare l'insieme su cui dobbiamo minimizzare la funzione obiettivo, ossia l'insieme

$$X = \{x \in \{0,1\}^n \mid Ax \geq e\}$$

in due insiemi, uno più interno, I, e uno più esterno, O. Inizialmente $I = X$ e $O = \varnothing$ Ad ogni passo restringeremo l'insieme I aggiungendo un nuovo vincolo del tipo indicato nella disequazione (9), e memorizzeremo (in z) il valore $x^* \in O$ che minimizza la funzione obiettivo su questo insieme. Quindi, ad ogni passo, il minimo valore della funzione obiettivo si raggiunge o in $x^* \in O$ o all'interno dell'insieme I.

Algoritmo:

1. Inizializziamo un contatore $k = 0$. Sia CP_0 il problema originale. Se una delle righe di A è il vettore nullo, terminiamo (non c'è soluzione ammissibile); altrimenti sia $z_(0) = e$ una prima copertura sicuramente funzionante.

2. Possiamo applicare al CP_k le semplificazioni introdotte nella sezione 5 [3].

3. Prendiamo una copertura prima J'_k per il CP_k, sia x^* la soluzione del CP associata come descritto nella sezione 7 e B la matrice di base relativa. Poniamo inoltre $z_{(k+1)} = z_{(k)}$ o $z_{(k+1)} = x^*$, quello dei due valori che minimizza la funzione obiettivo.

4. Calcoliamo il gradiente ridotto $c_N - c_B\hat{N}$, e sia Q l'insieme delle colonne dove esso è (strettamente) negativo. Se $Q = \varnothing$, allora la soluzione ottimale per il CP_k è x^*. In questo caso, l'algoritmo termina e la soluzione ottimale per il CP originale è $z_{(k+1)}$.

5. Se Q non è vuoto, consideriamo il vincolo aggiuntivo (piano secante) $\sum_{i \in Q} x_i \geq 1$ e sia CP_{k+1} il CP che si ottiene aggiungendo tale vincolo a quelli del CP_k. Incrementiamo k e ricominciamo dallo step 2.

L'algoritmo implementa l'idea sopra descritta: I è l'insieme delle soluzioni accettabili per il CP_k, ad ogni passo una soluzione ammissibile viene "scartata"

[3] Si noti che nei passaggi successivi al primo non è necessario verificare tutte le condizioni che portano alle semplificazioni, ma solamente quelle che potrebbero essere conseguenza dell'aggiunta di una riga alla matrice A.

da I e il suo valore viene confrontato con il massimo raggiunto in O (memorizzato in z_k).

L'algoritmo termina sicuramente in quanto le soluzioni accettabili sono al più 2^n e in ognuno dei passi ne eliminiamo da I almeno una; ogni singolo passo inoltre è abbastanza veloce perché richiede solo somme, moltiplicazioni e confronti (la necessità di invertire una matrice viene aggirata con la formula $g = c_N - c_B\hat{N}$). Tuttavia 2^n è un *upper bound* molto lasco: il vero numero di passi necessari dipende grandemente dalla scelta della copertura prima fatta al passo 3, scelta su cui siamo stati molto vaghi all'interno dell'algoritmo. Infatti, se fossimo ad esempio così fortunati da trovare immediatamente come copertura prima x^* la soluzione ottimale, allora l'algoritmo terminerebbe in un passo solo (si avrebbe subito gradiente ridotto positivo). Un modo efficiente di trovare le "candidate soluzioni" x^* velocizza enormemente l'algoritmo, ed è di questo che ci occuperemo nella sezione successiva.

10 Euristiche per la ricerca veloce di coperture prime a basso costo

La ricerca di una copertura prima si compone di due passi:

- Trovare una copertura con un basso valore della funzione obiettivo
- Estrarre una copertura prima da una copertura preesistente

Proponiamo (senza dimostrazione, poiché il loro funzionamento è abbastanza intuitivo) tre algoritmi, con efficienza e tempi computazionali ben diversi, per il primo passo:

1. Prendere $J' = J$, cioè includere tutti i pezzi disponibili, è sempre una copertura (a meno di casi insolubili del problema, quando una riga è il vettore nullo).

2. Se abbiamo una copertura per il CP_k, una copertura per il $\mathrm{CP}_k + 1$ si ottiene aggiungendo una qualunque colonna j con $j \in Q$ (cioè una colonna che soddisfi il vincolo aggiunto).

3. Possiamo risolvere il problema di programmazione lineare associato al CP:

$$\begin{cases} \min \langle c, x \rangle \\ Ax \geq e \\ x \geq 0 \in \mathbb{R}^n \end{cases}$$

   (ad esempio con l'algoritmo del simplesso). La soluzione ottimale di questo problema sarà un vettore di reali compresi tra 0 e 1 (è semplice notare che se fosse $x_k > 1$ potremmo ridurlo a $x_k = 1$ senza violare vincoli e riducendo il valore della funzione obiettivo); quindi, arrotondando per eccesso il vettore soluzione x (cioè prendendo $x^* = 0$ se $x = 0$ e $x^* = 1$ altrimenti) otteniamo una soluzione intera ammissibile per il CP, che solitamente è una soluzione di costo abbastanza basso.

A questi aggiungiamo due algoritmi per l'estrazione di una copertura prima, quello immediato e uno con un'euristica più raffinata:

1. Partiamo da una copertura J' e cerchiamo di eliminare man mano i pezzi in sovrappiù: esaminiamo una per una (ad esempio in ordine di costo decrescente) tutte le colonne $j \in J'$: ad ogni passo,se $J' \setminus \{j\}$ resta una copertura, poniamo $J' \leftarrow J' \setminus \{j\}$ e reiteriamo.

2. Sia sempre J' la copertura di partenza; questa volta partiremo da un insieme $S = \varnothing$ e aggiungeremo man mano i pezzi necessari con un algoritmo di tipo *greedy*: sia $Q = \{i | \sum_{j \in S} a_i j = 0$ l'insieme dei vincoli non soddisfatti dai pezzi di S; per ogni pezzo $j \in J' \setminus S$ calcoliamo il "costo medio per vincolo soddisfatto" $\frac{c_j}{\sum_{i \in Q} x_j}$ e aggiungiamo a S il pezzo che minimizza tale valore; reiteriamo fino a quando $Q = \varnothing$ e quindi S è una copertura.

Quest'ultimo algoritmo, partendo da $J' = J$, fornisce ottimi risultati in un tempo sensibilmente minore rispetto a quello della programmazione lineare, ed è quindi nella maggior parte dei casi la scelta migliore. Ad esempio, nel problema della figura 1 la soluzione trovata al primo tentativo con questo algoritmo è già quella ottimale.

11 Conclusioni e ulteriori sviluppi

L'algoritmo proposto rappresenta un metodo efficiente di risolvere il CP/PP ed è particolarmente adatto all'implementazione su computer in quanto la maggior parte dei calcoli si riducono in ultima analisi a operazioni dell'aritmetica binaria su vettori di $\{0,1\}^n$, operazioni che i computer gestiscono velocemente in modo nativo. L'algoritmo può essere inoltre convenientemente specializzato al PP o alle singole applicazioni con opportune modifiche alle riduzioni della sezione 5.

Come è stato possibile ridurre il PP al CP, è possibile attraverso opportune modifiche della funzione costo minimizzare quantità più complesse, come nell'esempio citato nella nota di pag. 2

Un problema simile al CP è il *problema di inclusione (IP)*, in cui si richiede di "appoggiare" all'interno della regione ammissibile il maggior numero possibile di pezzi, senza sovrapposizioni, eventualmente lasciando dei buchi.

Anche se con un incremento delle dimensioni del problema, è possibile ricondurre al PP (e quindi al CP) anche il problema di inclusione. L'idea è quella di porre $c_j = -1$ (o un diverso valore negativo, se alcuni pezzi sono più "vantaggiosi" di altri) per ognuno dei pezzi esistenti e aggiungere dei pezzi ausiliari che ricoprono ognuno una sola casella e hanno costo $c_j = 0$ (essi rappresentano le caselle lasciate vuote nella soluzione dell'IP). Nel ridurre poi il PP che ne deriva al CP si avrà cura di scegliere (nelle operazioni descritte nella sezione 4) un M abbastanza grande da rendere positivi tutti i costi.

Infine, esiste per il CP anche un algoritmo alternativo (qui non esaminato) usa le tecniche del *branch and bound* con buoni risultati in termini di tempo di computazione.

Riferimenti bibliografici

[1] XXX: libro fotocopiato da G.

[2] A. R. Brown. *Optimum Packaging and Depletion*. MacDonald – American Elesevier, 1971.

[3] http://linuz.sns.it/∼fvenez/pentamini.html (per alcune informazioni sul tassellamento di rettangoli con i pentamini)