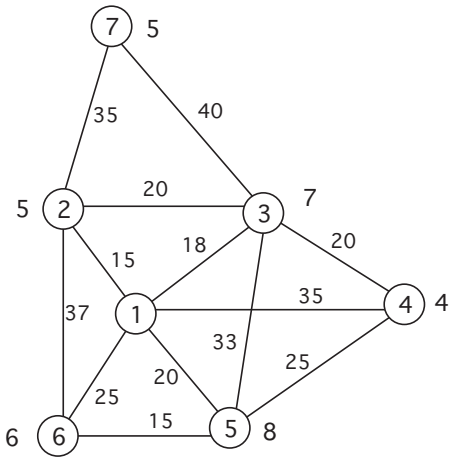
Esempio consegna merci



Il nodo 1 è il deposito. I numeri scritti accanto ai nodi rappresentano la quantità di merce da consegnare nel nodo. I numeri scritti accanto agli archi sono le distanze chilometriche fra le località. Pertanto si ha la seguente tabella delle distanze:

	1	2	3	4	5	6	7
1	–	15	18	35	20	25	50
2	15	–	20	40	35	37	35
3	18	20	–	20	33	43	40
4	35	40	20	–	25	40	60
5	20	35	33	25	–	15	70
6	25	37	43	40	15	–	72
7	50	35	40	60	70	72	–

inoltre sono disponibili due veicoli di capacità 21 entrambi.

Sviluppiamo due modelli, il primo diretto e il secondo con generazione di colonne.

Primo modello

$$x_e^k = \begin{cases} 1 & \text{se l'arco } e \text{ appartiene al circuito } k \\ 0 & \text{altrimenti} \end{cases}, \qquad z_i^k = \begin{cases} 1 & \text{se il nodo } i \text{ appartiene al circuito } k \\ 0 & \text{altrimenti} \end{cases}$$

$$\begin{aligned} \min \quad & \sum_k \sum_e c_e \, x_e^k \\ & \sum_{e \in \delta(i)} x_e^k \quad = 2 \, z_i^k \qquad i \in N, \; k \\ & \sum_{i \in N \setminus 1} w_i \, z_i^k \quad \leq C \qquad k \\ & \sum_{k=1}^{\min\{i-1,m\}} z_i^k = 1 \qquad i \in N \setminus 1 \\ & x_e \in \{0,1\}, \; z_i^k \in \{0,1\}, \; z_1^k = 1 \end{aligned} \tag{1}$$

Il vincolo $\sum_{e\in\delta(i)} x_e^k = 2\,z_i^k$ è il vincolo di grado nel nodo i per il circuito k, con la variante che il grado può essere 2 o 0 a seconda se il nodo è attraversato dal circuito k. Il vincolo $\sum_{i\in N\setminus 1} w_i\, z_i^k \le C$ è il vincolo di capacità per ogni veicolo. Infine il vincolo $\sum_{k=1}^{\min\{i-1,m\}} z_i^k = 1$ è un vincolo di assegnamento che impone ad ogni nodo di esser visitato da un veicolo. Si noti che a differenza dell'analogo vincolo $\sum_k z_i^k = 1$, quello scritto è più efficace perché elimina tutte le soluzioni equivalenti che si ottengono per semplice permutazione di circuiti.

Il modello presenta $(n\cdot m+n+m)$ vincoli (con n numero di nodi e m numero di veicoli). Una soluzione di (1) non è necessariamente un insieme di m circuiti passanti per il deposito. Come nel TSP possono essere presenti altri circuiti (oltre agli m previsti) e non passanti per il deposito. Per eliminarli bisogna aggiungere diseguaglianze di sottocircuito del tipo

$$\sum_{e\in\delta(S)} \sum_k x_e^k \ge 2$$

per ogni insieme S. Queste diseguaglianze vanno ovviamente generate nel momento in cui vengono violate dalla soluzione. Inoltre è utile aggiungere a (1) altre diseguaglianze che non alterano l'insieme delle soluzioni ammissibili, ma riducono l'insieme ammissibile rilassato e quindi sono efficaci nell'aumentare la limitazione inferiore. Le diseguaglianze sono

$$x_{ij}^k \le z_i^k, \qquad x_{ij}^k \le z_j^k \tag{2}$$

Si tratta di un consistente aumento di vincoli. Infatti se ne aggiungono $m\,n\,(n-1)$ e, se il problema è grande, può essere opportuno generarle solo se sono violate.

L'esempio è sufficientemente piccolo da poter inserire direttamente (2) in (1). Ecco il modello rilassato scritto in Lingo:

```
SETS:
   nodes/1..7/:w;
   arcs(nodes,nodes)|&1#LT#&2:c;
   tour/t1 t2/;
   npart(tour,nodes):z;
   apart(tour,arcs):x;
ENDSETS
DATA:
   w = 0 5 7 4 8 6 5;
   cap= 21;
   c= 15 18 35 20 25 50
         20 40 35 37 35
            20 33 43 40
               25 40 60
                  15 70
                     72;
ENDDATA
   min= @SUM(apart(k,i,j): c(i,j)*x(k,i,j));
   @FOR(tour(k):
      @FOR(nodes(i):
         @SUM(arcs(i,j): x(k,i,j) ) + @SUM(arcs(j,i): x(k,j,i) ) = 2* z(k,i) ));
   @FOR(apart(k,i,j): x(k,i,j) < z(k,i) ; x(k,i,j) < z(k,j) );
   @FOR(tour(k): @SUM(nodes(i) : w(i)*z(k,i) ) < cap );
   @FOR(nodes(i) | i#GT#1: @SUM(tour(k) | k#LT#i: z(k,i) ) = 1 );
   @FOR(tour(k): z(k,1) = 1 );
   @FOR(apart:@BND(0,x,1));
   @FOR(npart: @BND(0,z,1));
```

Risolvendolo si ottiene una soluzione frazionaria di valore 203.32 che fornisce quindi una limitazione inferiore di 204. Senza (2) si ottiene invece una soluzione frazionaria di valore 200. Quindi il miglioramento della limitazione inferiore è dell'ordine del 2%.

La soluzione intera ottima si ottiene con pochi nodi dell'albero branch-and-bound. La soluzione è data dai due tour:

$$1 \to 2 \to 7 \to 3 \to 4 \to 1, \qquad 1 \to 5 \to 6 \to 1$$

per una distanza totale di 205 chilometri.

Secondo modello

Dato un sottoinsieme p di $N' := N \backslash 1$ sia c_p il costo del circuito ottimo che parte e arriva a 1 attraversando esattamente una volta tutti i nodi di p. Sia a^p il vettore d'incidenza di p e sia $x_p \in \{0,1\}$ ad indicare se nella soluzione ottima i nodi di p vengono visitati dallo stesso veicolo. Sia $\mathcal{P}$ l'insieme di tutti i sottoinsiemi di N'. Il modello è

$$\begin{aligned} \min \quad & \sum_{p \in \mathcal{P}} c_p \, x_p \\ & \sum_{p \in \mathcal{P}} a_i^p \, x_p = 1 \qquad i \in N' \\ & \sum_{p \in \mathcal{P}} x_p \leq m \\ & x_p \in \{0,1\} \end{aligned} \tag{3}$$

Il modello (3) ha solo n vincoli, ma un numero esponenziale di colonne. Bisogna quindi generare un numero limitato di colonne utili a trovare una soluzione ottima. Il rilassamento di (3) è

$$\begin{aligned} \min \quad & \sum_{p \in \mathcal{P}} c_p \, x_p \\ & \sum_{p \in \mathcal{P}} a_i^p \, x_p = 1 \qquad i \in N' \\ & \sum_{p \in \mathcal{P}} x_p \leq m \\ & x_p \geq 0 \end{aligned} \tag{4}$$

Per generare le colonne di (4) è utile trasformare il modello di partizione di insiemi (3) in uno di copertura d'insiemi (cioè un nodo può appartenere a più sottoinsiemi). Questo non altera l'ottimo, in quanto una soluzione ammissibile per la partizione è anche ammissibile per la copertura e, se assumiamo la proprietà triangolare verificata, come avviene se i costi degli archi rappresentano distanze minime fra le città, nessuna soluzione ottima richiederà due circuiti passanti per lo stesso nodo. L'utilità di avere un modello di copertura è data dal fatto che le variabili duali sono positive in questo caso, mentre nel caso di partizione il loro segno non è noto a priori e i valori in uscita da un risolutore di PL possono presentare un segno alterato da operazioni preliminari fatte sulla matrice dei vincoli.

Quindi il modello su cui ci basiamo per generare colonne è:

$$\begin{aligned} \min \quad & \sum_{p \in \mathcal{P}} c_p \, x_p \\ & \sum_{p \in \mathcal{P}} a_i^p \, x_p \geq 1 \qquad i \in N' \\ & \sum_{p \in \mathcal{P}} x_p \leq m \\ & x_p \geq 0 \end{aligned} \tag{5}$$

Il problema duale di (5) è

$$\begin{aligned} \max \quad & \sum_{i \in N'} y_i - m\, w \\ & \sum_{i \in N'} a_i^p\, y_i - w \le c_p \qquad p \in \mathcal{P} \\ & y_i \ge 0,\ w \ge 0 \end{aligned} \tag{6}$$

L'ammissibilità duale per una assegnata variabile duale y è

$$c_p - \sum_{i \in N'} a_i^p\, y_i \ge -w \tag{7}$$

Siccome c_p è il costo del circuito ottimo che passa per i nodi di $p \cup 1$, verificare (7) corrisponde a risolvere un problema di *prize collecting TSP* (PCTSP) con vincolo di capacità. Nel PCTSP non c'è l'obbligo di visitare tutti i nodi (a parte il nodo di partenza 1) e, oltre ai costi sugli archi, sono presenti anche dei premi nei nodi, se questi sono visitati. Se non ci fossero i premi la soluzione ottima sarebbe un circuito nullo, ma la possibilità di ottenere dei premi di valore maggiore della distanza percorsa fa sì che la soluzione ottima possa essere un sottocircuito non nullo.

Inoltre in (7) solo sottoinsiemi di nodi ammissibili per il vincolo di capacità possono essere considerati. Quindi c'è una complicazione in più rispetto ad un normale PCTSP. Indichiamo con PSTSPC, il nuovo problema.

Allora per verificare (7) si deve risolvere un PCTSPC con premi dati dalle variabili duali y_i e la condizione d'ottimalità si ha quando la soluzione del PCTSP ha valore uguale a $-w$.

Si noti che per generare una colonna si deve risolvere un problema NP-hard. Sembra strano affrontare un problema NP-hard come quello originale, risolvendo molte volte un altro problema NP-hard. Se però l'algoritmo per il secondo problema ha un comportamento abbastanza più veloce del primo, l'approccio può dare dei risultati positivi.

Un modello per il PCTSPC è il seguente:

$$\begin{aligned} \min \quad & \sum_e c_e\, x_e - \sum_{i \in N'} y_i\, z_i \\ & \sum_{e \in \delta(i)} x_e \qquad\qquad = 2\, z_i \qquad i \in N \\ & \sum_{i \in N'} w_i\, z_i \qquad\qquad \le C \\ & x_e \in \{0,1\},\ z_i \in \{0,1\},\ z_1 = 1 \end{aligned} \tag{8}$$

Come nel precedente caso una soluzione intera può corrispondere a più di un circuito e quindi non essere ammissibile. Per eliminare i circuiti spurii bisogna ricorrere a diseguaglianze di sottocircuito, che però devono assumere una forma diversa in quanto non è a priori noto se alcuni nodi devono essere presenti nel circuito ottimo. Quindi bisogna far dipendere la diseguaglianza anche dal valore della variabile z_i. Per ogni nodo i e per ogni insieme S tale che $1 \in S$ e $i \notin S$ deve essere:

$$\sum_{e \in \delta(S)} x_e \ge 2\, z_i \tag{9}$$

La violazione di una diseguaglianza (9) si verifica risolvendo un massimo flusso fra 1 e i con capacità simmetriche $[-x_e, x_e]$. Se il massimo flusso è minore di $2\, z_i$ la diseguaglianza (9) va inserita in (8). Inoltre, anche in questo caso conviene rafforzare il modello rilassato con le diseguaglianze

$$x_{ij} \le z_i, \qquad x_{ij} \le z_j \tag{10}$$

da aggiungersi a (8). In questo modo il modello (8) generatore delle colonne di (4) ha $n+1+n\,(n-1) = O(n^2)$ vincoli (più eventuali vincoli di sottocircuito) che, pur non essendo pochi, sono minori dei $n\,m + n + m + m\,n\,(n-1) = O(m\,n^2)$ del modello (1). Con un elevato valore di m, l'aumento del numero di vincoli può risultare in un modello (1) intrattabile rispetto a (8).

Per risolvere (4) bisogna iniziare con un insieme di m circuiti che copra tutti i nodi. Potrebbe non essere banale trovare una tale soluzione iniziale. In fin dei conti bisogna risolvere un problema di bin packing e poi m problemi di TSP. Non volendo procedere in questo modo, si può sempre partire da un insieme arbitrario di m circuiti, non necessariamente ammissibili per il vincolo di capacità e non necessariamente di costo minimo. A questi circuiti bisogna assegnare un costo molto elevato in (4) per evitare che si presentino nella soluzione finale.

Per risolvere l'esempio possiamo iniziare con i due insiemi:

$$p_1 = \{2, 3, 4\}\,, \qquad p_2 = \{5, 6, 7\}$$

ai quali assegniamo i costi (arbitrari ed elevati)

$$c_1 = 1000, \qquad c_2 = 1000$$

Quindi il problema iniziale (4) è

$$\begin{aligned} \min \quad & 1000\,x_1 + 1000\,x_2 \\ & \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \geq 1 \\ & x_1 + x_2 \leq 2 \\ & x_1 \geq 0\, x_2 \geq 0 \end{aligned} \tag{11}$$

Risolvendo (11) si ottengono le seguenti variabili duali ottime:

$$y_2 = 1000,\ y_3 = 0,\ y_4 = 0,\ y_5 = 1000,\ y_6 = 0,\ y_7 = 0, \quad w = 0$$

Questi sono i premi da usare per risolvere il PCTSPC (8). Dai valori si può arguire che verrà cercato un circuito che usi i nodi 2 e 5. Infatti, risolvendo (8) si ottiene il circuito

$$p_3 = 1 \to 2 \to 5 \to 1$$

di lunghezza $c_3 = 70$. Quindi (4) diventa

$$\begin{aligned} \min \quad & 1000\,x_1 + 1000\,x_2 + 70\,x_3 \\ & \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \geq 1 \\ & x_1 + x_2 + x_3 \leq 2 \\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 \end{aligned} \tag{12}$$

Risolvendo (12) si ottengono le variabili duali ottime:

$$y_2 = 0,\ y_3 = 1000,\ y_4 = 0,\ y_5 = 70,\ y_6 = 930,\ y_7 = 0, \quad w = 0$$

Si ottiene il circuito

$$p_4 = 1 \to 3 \to 5 \to 6 \to 1$$

di lunghezza $c_4 = 91$. Continuando in questo modo si generano i circuiti

$$\begin{array}{lll} p_5 = 1 \to 2 \to 7 \to 4 \to 6 \to 1 & c_5 = 175 \\ p_6 = 1 \to 2 \to 3 \to 4 \to 1 & c_6 = 90 \\ p_7 = 1 \to 5 \to 4 \to 7 \to 1 & c_7 = 155 \\ p_8 = 1 \to 2 \to 7 \to 3 \to 1 & c_8 = 108 \\ p_9 = 1 \to 4 \to 5 \to 6 \to 1 & c_9 = 100 \\ p_{10} = 1 \to 2 \to 4 \to 5 \to 1 & c_{10} = 100 \\ p_{11} = 1 \to 3 \to 4 \to 5 \to 1 & c_{11} = 83 \\ p_{12} = 1 \to 2 \to 7 \to 3 \to 4 \to 1 & c_{12} = 145 \\ p_{13} = 1 \to 5 \to 6 \to 1 & c_{13} = 60 \\ p_{14} = 1 \to 2 \to 6 \to 5 \to 1 & c_{14} = 87 \end{array}$$

Introducendo il circuito p_5 si ottiene in (4) una soluzione diversa da $x_1 = x_2 = 1$ (e le altre variabili uguali a zero) e cioè $x_4 = x_5 = 1$ di valore 266. Dopo il circuito p_7 si ottiene una soluzione di valore 255.5. Dopo il circuito p_8 si ottiene una soluzione di valore 231.777. Dopo il circuito p_9 si ottiene una soluzione di valore 208. La soluzione ottima di valore 205 compare dopo l'introduzione del circuito p_{13}. Tuttavia, per dimostrarne l'ottimalità, bisogna ancora generare il circuito p_{14}. A questo punto il problema (8), con premi (cioè variabili duali ottime y_i)

$$y_2 = 27,\ y_3 = 10,\ y_4 = 37,\ y_5 = 36,\ y_6 = 24,\ y_7 = 71$$

fornisce nuovamente il circuito p_{13} che ha lunghezza 60 e premio anche 60. Siccome $w = 0$, Quindi si è raggiunta l'ottimalità.

Il modello Lingo di (5) è (alla fine dell'introduzione di tutti i circuiti)

```
SETS:
   nodes/1..6/:w;
   tours/t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14/:x,c;
   mat(tours,nodes)/
      t1 1 t1 2 t1 3
      t2 4 t2 5 t2 6
      t3 1 t3 4
      t4 2 t4 4 t4 5
      t5 1 t5 6 t5 3 t5 5
      t6 1 t6 2 t6 3
      t7 4 t7 3 t7 6
      t8 1 t8 6 t8 2
      t9 3 t9 4 t9 5
      t10 1 t10 3 t10 4
      t11 2 t11 3 t11 4
      t12 1 t12 6 t12 2 t12 3
```

```
    t13 4 t13 5
    t14 1 t14 4 t14 5;
ENDSETS
DATA:
  c= 1000 1000 70 91 175 90 155 108 100 100 83 145 60 87;
  m=2;
ENDDATA
  min= @SUM(tours: c*x);
  @FOR(nodes(i) : [y] @SUM(mat(j,i) : x(j) ) > 1 );
  [w] @SUM(tours : x ) < m ;
```

mentre il modello Lingo del generatore di colonne (8) è

```
SETS:
  nodes/1..7/:w,z,y;
  arcs(nodes,nodes)|&1#LT#&2:c,x;
ENDSETS
DATA:
  w = 0 5 7 4 8 6 5;
  cap= 21;
  c= 15 18 35 20 25 50
       20 40 35 37 35
          20 33 43 40
             25 40 60
                15 70
                   72;
  y=0 27 10 37 36 24 71;
ENDDATA
  L= @SUM(arcs(i,j): c(i,j)*x(i,j));
  prize=@SUM(nodes(i): z(i)*y(i));
  min = L-prize;
  @FOR(nodes(i):
    @SUM(arcs(i,j): x(i,j) ) + @SUM(arcs(j,i): x(j,i) ) = 2* z(i));
  @FOR(arcs(i,j): x(i,j) < z(i) ; x(i,j) < z(j) );
  @SUM(nodes(i) : w(i)*z(i) ) < cap ;
  z(1)=1;
  @FOR(arcs:@BND(0,x,1);@GIN(x));
  @FOR(nodes:@BND(0,z,1);@GIN(z));
```