1 Lezione del 23-10-24

Prendiamo in considerazione il poliedro generico di ILP:

$$\begin{cases} Ax \leq b \\ x \in \mathbb{Z}^n \end{cases}$$

Facciamo un riassunto sulle tecniche greedy per il ricavo di una soluzione ammissibile (che danno un limite inferiore v_I) e sul ricavo del rilassato (che dà un limite superiore v_S), con riferimento alla ILP. Bisogna notare che un algoritmo greedy da vincolo inferiore su problemi di massimo, e superiore altrimenti. Allo stesso modo, il rilassato dà vincolo superiore su problemi di massimo, e inferiore altrimenti.

1.0.1 Algoritmi per valutazioni inferiori

- **Zaino:** abbiamo visto gli **algoritmi dei rendimenti** per il caricamento di problemi di zaino in ILP, cioè per uno zaino di vettore valore v e peso p, con peso massimo P, si calcolano i rendimenti:
$$r_i = \frac{v_i}{p_i}$$
e si selezionano uno o più elementi di rendimento massimo (dipende anche dal tipo booleano o meno di problema), saturando fino a frazione se si è nel rilassato, e fino a intero in caso contrario;
- **Bin-packing:** per la ricerca di soluzioni ammissibili dei problemi di bin-packing abbiamo visto tre algoritmi di ricerca, **next-fit decreasing**, **first-fit decreasing** e **best-fit decreasing**;
- **Algoritmo delle toppe:** l'algoritmo delle toppe per i problemi TSP asimmetrici rappresenta ancora un metodo euristico per il calcolo di cicli hamiltoniani a costo minimo.

1.0.2 Algoritmi per valutazioni superiori

Abbiamo introdotto il concetto di rilassamento di vincoli, in modo da trovare un sovrainsieme della regione ammissibile di ILP da dove è più facile ricavare una soluzione ammissibile. In particolare, la tecnica più immediata è quella di rimuovere il vincolo di interezza ($x \in \mathbb{Z}^n$) o booleano ($x \in \{0,1\}^n$).

1.1 TSP simmetrico

Veniamo quindi ai problemi di TSP simmetrico, cioè dove la tabella dei costi è simmetrica, ovvero in forma:

$$c = \begin{pmatrix} 27 & 23 & 24 & 26 \\ - & 21 & 32 & 33 \\ - & - & 41 & 42 \\ - & - & - & 47 \end{pmatrix}$$

dove non si riporta la parte in basso a sinistra in quanto è la specchiata della matrice triangolare in alto a destra.

Siccome questo è un caso particolare del TSP, possiamo effettivamente usare tutti i metodi studiati per il TSP asimmetrico. Vediamo però che ci sono delle semplificazioni particolari che possiamo fare sul problema.

Innanzitutto, visto che possiamo prendere gli archi come non orientati, le variabili possibili sono dimezzano. Se nello scorso problema avevamo $5 \cdot 5 = 25$ variabili meno 5 della diagonale, ergo $n^2 - n$, adesso ne abbiamo soltanto $\frac{n^2-n}{2}$.

Abbiamo poi che i vincoli si presentano in forma:

$$\begin{cases} \min c^T x \\ \sum\limits_{h,i \in A} x_{hi} + \sum\limits_{i,k \in A} = 2, \quad \forall i \in N \\ \sum\limits_{i \in S, j \notin S} x_{ij} + \sum\limits_{i \notin S, j \in S} x_{ij} \geq 1, \quad \forall S \subset N, \quad 1 \leq |S| \leq \left\lceil \frac{|N|}{2} \right\rceil \end{cases}$$

La prima serie di vincoli rappresenta il fatto che ogni nodo è parte di al massimo due archi, ergo uno uscente e entrante (anche se questa definizione non ha molto significato per grafi non orientati). La seconda serie rappresenta invece i vincoli di connessione, che in questo caso prendiamo in entrambe le direzioni, e che limitiamo in dimensioni di sottoinsieme alla cardinalità $|N|$ fratto 2, al limite approssimata all'eccesso, in quanto il vincolo per S vale anche per $N \setminus S$.

1.1.1 Valutazioni inferiori e superiori

Un'approccio greedy alla soluzione, che fornisce un vincolo superiore v_S (siamo in minimo), è quello di scegliere un nodo ad arbitrio e proseguire da lì in poi scegliendo il nodo con $\min(c_{xj})$ di adiacenza, ovvero:

Algoritmo 1 del nodo vicino

Input: un insieme N di nodi
Output: una valutazione superiore v_S del TSP simmetrico
Parti da un nodo ad arbitrio;
ciclo:
Scegli il nodo adiacente più vicino e unifica
if ci sono altri nodi **then**
  Torna a ciclo
end if

Per il calcolo di una valutazione inferiore avremo bisogno di un rilassamento. Scegliamo di rilassare i vincoli sul grado di tutti i nodi tranne uno, detto k:

$$\begin{cases} \min c^T x \\ \sum\limits_{h,r \in A} x_{hi} + \sum\limits_{r,k \in A} = 2, \quad \forall i \in N \\ \sum\limits_{i \in S, j \notin S} x_{ij} + \sum\limits_{i \notin S, j \in S} x_{ij} \geq 1, \quad \forall S \subset N \setminus \{r\}, \quad 1 \leq |S| \leq \left\lceil \frac{|N|}{2} \right\rceil \end{cases}$$

Diamo quindi la seguente definizione:

Definizione 1.1: K-albero

Scelto un nodo k, chiamiamo k-albero un insieme di n archi dove $n - 2$ archi formano un albero di copertura sul sottografo formato dai nodi $N \setminus \{k\}$, e 2 archi incidono sul nodo k.

In sostanza, un k-albero è un **albero di copertura** con un ciclo. Bisogna notare che questa definizione di k-albero differisce da quella comunemente usata sia in informatica (albero con ramificazione k) che in teoria dei grafi (un altro tipo di grafo non diretto). Inoltre, si trova anche sotto altri nomi, tra cui r-albero, che però è ancora conflitto con gli r-grafi informatici (che sono un tipo di albero per l'organizzazione di informazione spaziale).

Tolta quest'ultima precisazione, abbiamo che ciò che vogliamo è un k-albero minimo. Ricordiamo quindi il seguente algoritmo:

1.1.2 Algoritmo di Kruskal

Possiamo usare l'**algoritmo di Kruskal** per trovare l'albero di copertura minimale per un insieme N di nodi. Assumendo grafi connessi, si può formulare come:

Algoritmo 2 di Kruskal

Input: un insieme N di nodi
Output: un albero di copertura minimale di N
ciclo:
Ordina gli archi in maniera crescente rispetto al peso
Scegli il primo arco.
if aggiungere l'arco non crea cicli **then**
  Aggiungi l'arco e unifica le componenti
else
  Scegli il prossimo arco e riprova
end if
if non hai finito gli archi **then**
  Vai a ciclo
end if

Questo algoritmo trova sempre una soluzione, cioè un albero di copertura minimale, e visto che sceglie sempre archi a costo minimo, trova sempre la soluzione ottima.

Ritornando al problema dei k-alberi minimi, possiamo finalmente presentare il seguente algoritmo per il calcolo di un k-albero di costo minimo su qualsiasi nodo k:

Algoritmo 3 del k-albero

Input: un insieme N nodi, di cui $k \in N$
Output: un k-albero di costo minimo
Trova l'albero di copertura minimale C_H con Kruskal del sottografo $N \setminus \{k\}$
Connetti k a C_H attraverso i due nodi a costo minimo

Possiamo quindi dire perchè abbiamo introdotto il concetto di k-albero: il problema rilassato posto prima, dove si erano rilassate le cardinalità degli archi su tutti i nodi tranne k, ha come soluzione il k-albero a costo minimo. Trovando questo k-albero, abbiamo una valutazione superiore del problema di TSP simmetrico di partenza.