1 Lezione del 24-10-24

1.0.1 Riassunto TSP simmetrico

Avevamo quindi posto problemi con vincoli in forma:

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

cioè dove si poneva la somma dei nodi entranti e uscenti (i vincoli *di grado*) da j come $= 2$. Visto che j era l'unico nodo su cui era imposto il vincolo, si aveva che la soluzione del problema era il j-albero a costo minimo.

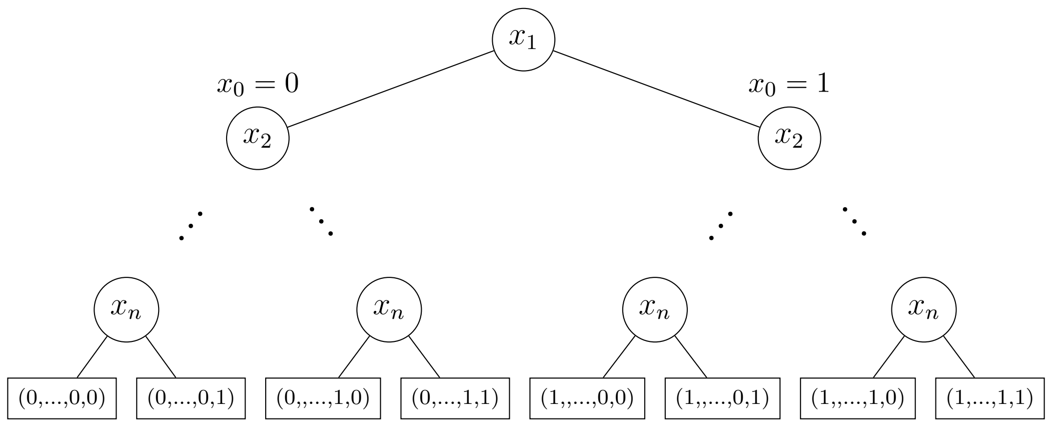
Inoltre, si aveva che i vincoli sulla terza erano quelli *di connessione*, che si cercavano solo su cardinalità dei sottoinsiemi $= \left\lceil \frac{|N|}{2} \right\rceil$, in quanto i vincoli per S valevano anche per $N \setminus S$.

1.1 Branch and bound

Avevamo visto l'algoritmo dei piani di taglio di Gomory per il calcolo di soluzioni approssimate di problemi ILP. Presentiamo adesso un altro metodo, detto **branch and bound**. Il metodo più naive che possiamo adottare per risolvere un problema in forma:

$$\begin{cases} \max c^T x \\ Ax \leq b \\ x \in \{0, 1\}^n \end{cases}$$

è quello di enumerare tutti le possibili soluzioni ammissibili $\in \{0,1\}^n$, costruendo il cosiddetto **albero di enumerazione**. Scegliamo quindi una variabile, x_1, e costruiamo l'albero:



Abbiamo che il calcolo di ogni nodo dell'albero non è effettivamente necessario. Chiamiamo quindi **problemi** P_{ij} ogni nodo dell albero, con i che seleziona il livello e j il fratello, da sinstra verso destra.

Stabilita una valutazione superiore e inferiore di ogni problema, che chiameremo $v_S(P)$ e $v_I(P)$, abbiamo:

$$v_I(P) \leq v(P) \leq v_S(P)$$

Possiamo usare queste valutazioni per stabilire **regole di taglio** che ci permettano di tagliare (si dice anche *visitare implicitamente*) un'intero sottoalbero a partire da un certo nodo P_{ij}. Queste regole di taglio assicurano, essenzialmente, che ogni sottoproblema $P_{i+k,j'}$ figlio di P_{ij} non contiene l'ottimo, e quindi si può saltare.

Vediamo quindi le regole di taglio che possiamo adottare. Mantendendo un'ottimo corrente x, abbiamo che calcolato un nuovo P_{ij} per enumerazione:

1. $P_{ij} = \emptyset$ significa che per un certo nodo P_{ij} possiamo tagliare tutti i problemi che istanziano le i variabili di P_{ij}, ergo tutti i suoi nodi figli;

2. Calcolo di $v_S(P_{ij})$. Se $v_S(P_{ij}) < v_I(P)$ del problema, allora posso scartare il sottoalbero: non troverò soluzioni migliori scendendovi;

3. $v_S(P_{ij}) > v_I(P)$, e l'$\bar{x}$ dove si ha tale v_S è ammissibile per P, allora si prende $\bar{x}$ come nuovo x e si fa una visita implicità di P_{ij}: questo perchè un suo sottoproblema non potrà darci di meglio (più scendiamo nell'albero, più stringiamo i vincoli, ergo $P_{i+k,j'}$ figlio di P_{ij} ha $v_S(P_{i+k,j'}) \leq v_S(P_{ij})$).