1 Lezione del 16-10-24

1.1 Problema di impacchettamento

Problema 1.1: Impacchettamento

Poniamo di avere 6 file, contenenti registrazioni di tutta la musica di Mozart, con il seguente ingombro in gigabyte:

	Arie	Opere	Concerti	Sinfonie	Sonate	Messe
p	3	6	5	4	4	8

Vogliamo trovare il numero minimo di dischi rigidi di dimensione $P = 10$ per archiviare tutti di questi file.

Chiamiamo questi problemi anche problemi di *bin-packing*. Rappresentiamo la soluzione come una matrice di adiacenza:

$$x_{ij} = \begin{cases} 0 \\ 1 \end{cases}$$

dove $i \in \{1, ..., p\}$ rappresenta il contenitore e $j \in \{1, ..., 6\}$ l'oggetto che vi inseriamo. Conviene trovare prima una stima superiore per il numero di contenitori p, attraverso un'opportuno algoritmo greedy.

Un'algoritmo banale può essere quello di riempire finché è possibile il primo contenitore, cioè finche si hanno oggetti che entrano nello spazio libero (detto *sfrido*) del contenitore. Una volta che questa ipotesi è violata, si prende un'altro contenitore, e così via.

Si ricava quindi questa valutazione superiore V_S. A questo punto si può porre:

$$\begin{cases} x_{11} + x_{21} + x_{31} + x_{41} = 1 \\ x_{12} + x_{22} + x_{32} + x_{42} = 1 \\ ... \\ x_{16} + x_{26} + x_{36} + x_{46} = 1 \end{cases}$$

cioè vogliamo prendere uno e uno solo di tutti gli oggetti, disposti fra i $V_S = 4$ contenitori di valutazione superiore.

Visto che non siamo sicuri di dover prendere tutti i contenitori, dovremo introdurre una variabile y_i per ognuno di essi:

$$y_i = \begin{cases} 0 \\ 1 \end{cases}$$

Infine vogliamo inserire la dimensione di ogni contenitore, $P = 10$, nel problema, sulla base dei pesi p_i di ogni oggetto:

$$\begin{cases} p_1x_{11} + p_2x_{12} + p_3x_{13} + p_4x_{14} + p_5x_{15} + p_6x_{16} \leq 10y_1 \\ p_1x_{21} + p_2x_{22} + p_3x_{23} + p_4x_{24} + p_5x_{25} + p_6x_{26} \leq 10y_2 \\ ... \\ p_1x_{41} + p_2x_{42} + p_3x_{43} + p_4x_{44} + p_5x_{45} + p_6x_{46} \leq 10y_4 \end{cases}$$

Combinando quanto posto finora, otteniamo il problema completo:

$$\begin{cases} \min(y_1 + y_2 + y_3 + y_4) \\ x_{11} + x_{21} + x_{31} + x_{41} = 1 \\ x_{12} + x_{22} + x_{32} + x_{42} = 1 \\ ... \\ x_{16} + x_{26} + x_{36} + x_{46} = 1 \\ p_1x_{11} + p_2x_{12} + p_3x_{13} + p_4x_{14} + p_5x_{15} + p_6x_{16} \leq 10y_1 \\ p_1x_{21} + p_2x_{22} + p_3x_{23} + p_4x_{24} + p_5x_{25} + p_6x_{26} \leq 10y_2 \\ ... \\ p_1x_{41} + p_2x_{42} + p_3x_{43} + p_4x_{44} + p_5x_{45} + p_6x_{46} \leq 10y_4 \\ x_i \in \{0, 1\} \\ y_i \in \{0, 1\} \end{cases}$$

dove minimizziamo gli y_1 cercando di usare meno contenitori possibile. Questo è un problema di ILP. Cerchiamo quindi la valutazione inferiore V_I e la superiore V_S.

Possiamo dare una stima inferiore attraverso la formula:

$$V_I = \left\lceil \frac{\sum_{j=1}^{6} p_i}{P} \right\rceil$$

cioè il peso di tutti gli oggetti fratto le dimensioni dei contenitori, arrotondato per eccesso, che è il minimo numero di contenitori possibile per contenere tutti gli oggetti.

Per il calcolo della stima superiore, invece, avevamo presentato un algoritmo greedy. In verità sono ci altre (e più intelligenti) strade che possiamo prendere:

- **Next-fit decreasing:** essenzialmente l'algoritmo presentato, dove si p:

Algoritmo 1 next-fit decreasing per impachettamento

```
Input: un problema di impacchettamento
Output: una soluzione ammissibile
while ci sono ancora oggetti do
    Prendi il prossimo oggetto
    if entra nel contenitore then
        Inseriscilo nel contenitore
    else
        Prendi un'altro contenitore e inseriscici l'oggetto
    end if
end while
```

- **First-fit decreasing:** analogo al next-fit, ma con la differenza che per ogni oggetto si considerano tutti i contenitori:

Algoritmo 2 first-fit decreasing per impachettamento

Input: un problema di impacchettamento
Output: una soluzione ammissibile
while ci sono ancora oggetti **do**
  Prendi il prossimo oggetto
  if l'oggetto entra in uno dei contenitori presi finora **then**
    Inseriscilo nel contenitore
  else
    Prendi un'altro contenitore e inseriscici l'oggetto
  end if
end while

- **Best-fit decreasing:** è una variante del first-fit che prende sempre i contenitori con sfrido massimo, cioè cerca di riempire i contenitori con meno spazio disponibile (cioè di trovare l'"incastro" migliore per l'oggetto):

Algoritmo 3 best-fit decreasing per impachettamento

Input: un problema di impacchettamento
Output: una soluzione ammissibile
while ci sono ancora oggetti **do**
  Prendi il prossimo oggetto
  if l'oggetto entra in uno dei contenitori, ordinati per sfrido decrescente, presi finora **then**
    Inseriscilo nel contenitore
  else
    Prendi un'altro contenitore e inseriscici l'oggetto
  end if
end while