

$H_n = \text{numero armonico}$   
 $H_n = 1 + 1/2 + 1/3 + \dots + 1/n$

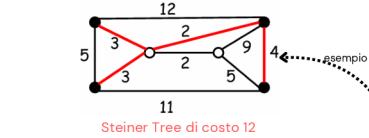
#### MINIMUM SET COVER PROBLEM:

- Input: Un insieme universale  $U$  di  $n$  elementi e una collezione  $S=\{S_1, S_2, \dots, S_k\}$  di sottoinsiemi di  $U$ . Ogni sottoinsieme  $S$  ha un costo  $c(S) > 0$ .
- Obiettivo: Selezionare una sotto-collezione  $C \subseteq S$  tale che l'unione di tutti gli insiemi in  $C$  copri  $U$  (cioè ogni elemento di  $U$  è contenuto in almeno un insieme di  $C$ ) e che la somma dei costi  $c(S)$  sia minima.

**Algoritmo** che lo risolve: strategia di tipo greedy che sceglie l'insieme di costo più conveniente finché tutti gli elementi non sono coperti. Questo algoritmo è una  $H_n$ -apx per il problema del minimum set cover, dove  $H_n = 1 + 1/2 + \dots + 1/n$ .

- DEF (Matching):** Dato un grafo  $G=(V,E)$ , un sottoinsieme degli archi  $M$  contenuto in  $E$  è un matching se non ci sono due archi in  $M$  che condividono uno stesso nodo (endpoint).
- DEF (Matching massimale):** Un matching  $M$  contenuto in  $E$  si dice massimale se per ogni arco e contenuto in  $E \setminus M$ ,  $M \cup \{e\}$  non è un matching.

luigi riggi



#### PASSAGGIO AD ISTANZA METRICA:

Una tecnica comune consiste nel trasformare il problema generale in una sua variante metrica. Il vantaggio di passare al caso metrico è che l'istanza cattura "l'essenza" della difficoltà del problema.

#### Nel Metric Steiner Tree Problem:

- Grafo Completo:** Si costruisce un grafo completo  $G'=(V,E')$  dove, per ogni coppia di vertici  $(u,v)$ , il costo  $c'(u,v)$  è definito come il costo del cammino più breve tra  $u$  e  $v$  nel grafo originale  $G$ .
  - Algoritmo** che lo risolve: l'algoritmo per risolvere il Metric Steiner Tree Problem, che garantisce un fattore di approssimazione pari a 2, si basa sui seguenti passaggi:
    - Selezione dei Vertici Richiesti: Si considerano solo i vertici dell'insieme  $R$ .
    - Sottografo Indotto: Si costruisce il sottografo di  $G'$  indotto da  $R$  (cioè, si considerano solo le connessioni tra i vertici richiesti).
    - Calcolo del Minimum Spanning Tree (MST): Si calcola l'MST sul sottografo indotto da  $R$ .
    - Restituzione della Soluzione: L'MST così calcolato viene restituito come soluzione approssimata.
- Idea Analitica: Se  $T^*$  è un albero di Steiner ottimale di costo  $OPT$ , duplicando tutti gli archi di  $T^*$  si ottiene un grafo Euleriano di costo  $2 \cdot OPT$ . Da questo si estra il tour Euleriano e, applicando la tecnica dello shortcut (che sfrutta la disugualanza triangolare), si ottiene un ciclo che passa solo per i vertici richiesti e il cui costo è al più  $2 \cdot OPT$ . Siccome l'MST ha costo non superiore a questo ciclo, il costo dell'MST è al più  $2 \cdot OPT$ .
- L'algoritmo è una **2-apx** per il minimum steiner tree problem.

#### RIPASSO MST

Un Minimum Spanning Tree (MST) di un grafo connesso, non orientato e pesato è un sottoinsieme di archi che:

- connette tutti i vertici del grafo,
  - minimizza la somma dei pesi degli archi,
  - non contiene cicli (è un albero).
- Esistono due algoritmi principali per trovare un MST in modo efficiente:
1. Algoritmo di Prim in tempo  $O(m+n\log n)$  (costruisce MST incrementalmente, partendo da un vertice e aggiungendo sempre l'arco di peso minimo che collega un nuovo vertice);
  2. Algoritmo di Kruskal in tempo  $O(m\log n)$  (ordina tutti gli archi per peso e li aggiunge al MST solo se non formano un ciclo).

luigi riggi

**INTRO**  
 Un algoritmo di  $\alpha$ -approssimazione per un problema di ottimizzazione è un algoritmo di tempo polinomiale che per tutte le istanze del problema produce una soluzione il cui valore è entro un certo fattore  $\alpha$  rispetto al valore di una soluzione ottimale. Questo approccio è ampiamente utilizzato per problemi NP-HARD.

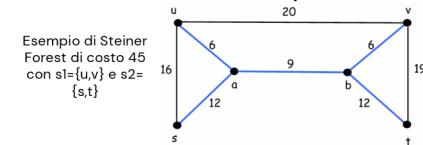
## ALGORITMI DI APPROSSIMAZIONE

**SCHEMA PRIMALE DUALE**  
 L'idea di alto livello dell'approccio è la seguente:  
 1. L'algoritmo inizia con una soluzione primale intera non ammessa e una soluzione doppia fittibile.  
 2. Iterativamente si migliora la soluzione doppia e si migliora la fattibilità della soluzione primale intera. Questo finché non si ottiene una soluzione primale intera fattibile.  
 3. Analisi: dimostrare la garanzia di approssimazione utilizzando il valore della soluzione doppia come limite inferiore.

**ALGORITMO PER SET COVER CHE PASSA PER IL PRIMALE DUALE (e non per il rounding come in lezione 3)**  
 Funzionamento algoritmo:  
 1.cominciamo con soluzione doppia ammessa (tutti  $y=0$ ) e soluzione non ammessa (tutti  $x=0$ )  
 2.finché tutti gli elementi sono coperti do: prendere un elemento e' qualsiasi non ancora coperto ed alzare variabili  $y$  (migliorando valore doppia)  
 finché qualche insieme non diventa tight  
 3.a questo punto uno o più insiemis sono tight, prendere tutti insiemis tight e mettere a 1 la corrispondente variabile  
 4.quindi otteniamo e restituiamo il set cover ammesso  $x$   
 THM: questo algoritmo è una **f-apx** per Set Cover

#### RISOLUZIONE DELLO STEINER FOREST PROBLEM TRAMITE SCHEMA PRIMALE DUALE

- Problema: prende in input un grafo non orientato  $G=(V,E)$  con costi degli archi positivi, e anche una collezione di sottoinsiemi disgiunti di  $V$  chiamata  $S=\{S_1, \dots, S_k\}$ . Una soluzione ammessa è una foresta  $F$  in cui ogni coppia di vertici appartenenti allo stesso insieme  $S_i$  è collegata. La misura da minimizzare è il costo di  $F$ , ossia la somma dei costi degli archi appartenenti ad  $F$ .
- Algoritmo funziona nel modo seguente:  
 1.all'inizio non ho nemmeno un arco preso nella foresta, quindi soluzione primale intera non ammessa. Inoltre soluzione doppia ammessa con tutti gli  $y=0$ .  
 2.Finché c'è un insieme non soddisfatto (finché sol. non ammessa) prendere tutti insiemis attivi di  $S$  (uno per ogni componente connessa che ha  $f(s)=1$ ) e alzare le corrispondenti variabili duali tutte insieme finché un arco non diventa tight, a questo punto ci fermiamo e mettiamo questo arco dentro  $F$ . Ripetiamo...  
 3.(pruning) La soluzione ottenuta dalla fase 2 è ammessa ma potrebbe avere costo troppo alto, rimuovere quindi da  $F$  tutti gli archi che sono ridondanti (ossia partendo da  $F$  ammesso, se togliendo l'arco e' F rimane ammesso allora lo togliamo). Otteniamo soluzione  $F'$  che è la soluzione cercata (**2-apx**).



#### FORMULAZIONE IN PROGRAMMAZIONE LINEARE INTERA DEL PROBLEMA MINIMUM SET COVER

Si introduce una variabile binaria  $x_S$  per ogni sottoinsieme  $S$  dove:

- $x_S=1$  se  $S$  è scelto nella soluzione,
- $x_S=0$  altrimenti.

L'ILP diventa:

$$\begin{aligned} \text{minimizza: } & \sum_{S \in S} c(S) \cdot x_S \\ \text{vincoli: } & \sum_{S \in S} x_S \geq 1 \quad \text{per ogni } e \in U, \\ & x_S \in \{0, 1\} \quad \text{per ogni } S \in S. \end{aligned}$$

Poi si rilassa il vincolo che  $x_S$  debba essere binario, permettendo valori frazionari. Così si ottiene un problema di programmazione lineare che può essere risolto in tempo polinomiale.

**TEOREMA DUALITA' DEBOLE**  
 Qualsiasi soluzione ammessa per il duale fornisce un limite inferiore al costo ottimo del problema primale.

**NB:** Ogni soluzione ammessa del duale è un lower bound alla soluzione ottima del primale.

**NB:** Ogni soluzione ammessa del primale è un upper bound alla soluzione ottimale del duale.

#### ALGORITMO LP-ROUNDING

E' un algoritmo di approssimazione per set cover, che passando per la programmazione lineare risolve il problema con una **f-aprossimazione**.

Funzionamento:

- 1.Trova una soluzione ottimale per il rilassamento LP
- 2.Sceglie tutti gli insiemis  $S$  per i quali  $x_S \geq 1/f$

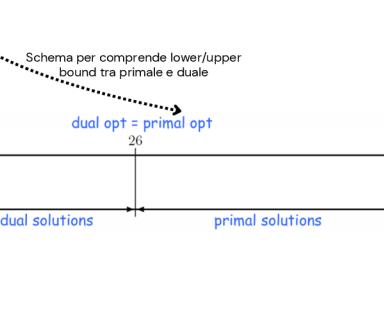
Quindi, tramite LP-ROUNDING si arrotonda la soluzione LP scegliendo quegli insiemis il cui valore è almeno  $1/f$ , garantendo un costo al più  $f$  volte l'ottimo.

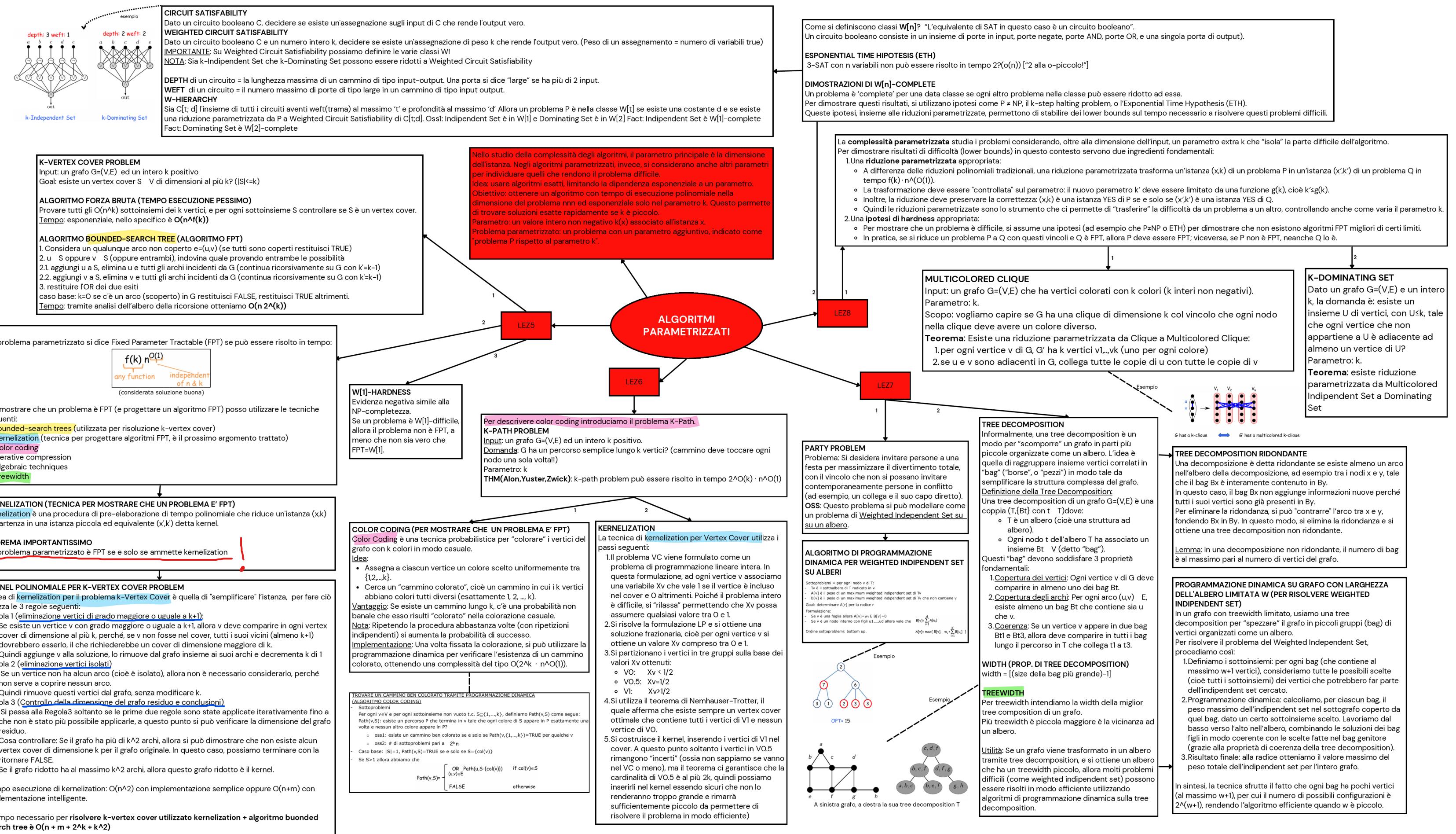
#### WEIGHTED VERTEX COVER PROBLEM

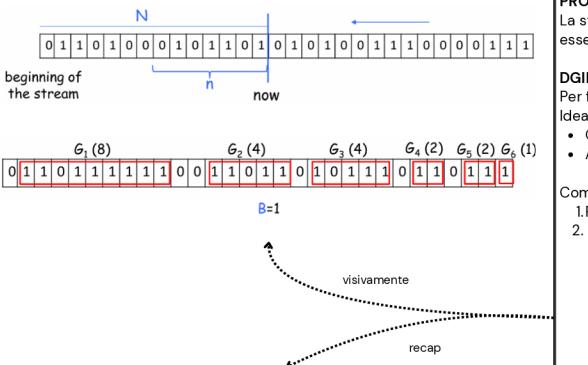
E' un caso speciale del set cover in cui scegliamo i vertici per coprire gli archi.

E' utilizzato come esempio in cui  $f=2$ , e la tecnica produce un algoritmo 2-apx.

**SET COVER TRAMITE DUAL-FITTING**  
 Funzionamento:  
 1.prendiamo il problema di programmazione lineare intera  
 2.lo rilassiamo  
 3.scriviamo il duale corrispondente a partire dal problema di programmazione lineare







- DGIM data structure:**
- quality:  $O(\epsilon^{-1} \log^2 N)$  approximated answers (for any  $\epsilon > 0$ )
  - size:  $O(\epsilon^{-1} \log^2 N)$  bits
  - update time:  $O(\log N)$
  - query time:  $O(\epsilon^{-1} \log n)$

