KAIST, School of Computing, Spring 2024
Algorithms for NP-hard problems (CS492)
Lecture: Eunjung KIM

Scribed By: Eunjung KIM
Lecture #22
9 May 2024

Contents

1 Greedy algorithm for weighted SET COVER and MAX COVERAGE

We discuss how to handle weights for SET COVER and MAX COVERAGE.

1.1 The case of WEIGHTED MAX COVERAGE

Notice: the content of this subsection is verbatim the same as the one for unweighted MAX COVERAGE. The content is repeated here to verify and emphasize that the same greedy procedure transfers to the weighted setting with minimal adaptation.

WEIGHTED MAX COVERAGE
Instance: a universe U with a weight function $\omega : U \to Q^+$, a collection $\mathcal{S}$ of subsets of U, a positive integer k.
Goal: Find a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ consisting of k sets which maximizes $\sum_{e \in \bigcup_{S \in \mathcal{S}'} S} \omega(e)$

For WEIGHTED MAX COVERAGE, almost the same greedy procedure for the unweighted works with minor modification. Instead of choosing a set which maximizes the *number* of freshly covered elements, we choose a set which maximizes the *weight sum* of freshly covered elements. For any subset $X \subseteq U$ of elements, we write $\omega(X)$ to denote $\sum_{e \in X} \omega(e)$.

Algorithm 1 Algorithm for WEIGHTED MAX COVERAGE
```
1: procedure MaxCoverage($U, \omega, \mathcal{S}, k$)
2:     Mark every element of $U$ as uncovered.
3:     $\mathcal{S}' \leftarrow \emptyset$ and $C \leftarrow \bigcup_{S \in \mathcal{S}'} S$
4:     while $U$ has an uncovered element and $|\mathcal{S}'| < k$ do
5:         Select $X \in \mathcal{S} \setminus \mathcal{S}'$ such that $\omega(X \setminus C)$ is as large as possible.
6:         $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{X\}$
7:         Mark every element of $X$ covered; $C \leftarrow C \cup X$.
8:     return $\mathcal{S}'$.
```

- Let $\mathcal{S}^*$ be an optimal set cover and let $U^* := \bigcup_{S \in \mathcal{S}^*} S$.
- Let $X_1, \ldots, X_i, \ldots, X_k$ be the chosen sets by the greedy algorithm.
- Let $g_t := \omega(U^*) - \omega(\bigcup_{i=1}^{t} X_i)$, i.e. the gap between the weight sum of elements that an optimal solution can cover and what the greedy solution covers at t.

The key observation is that for every $t+1$ before the termination of the WHILE-loop, the elements in U_t^* can be covered by k sets (e.g. by an optimal set cover), so there exists a set $Y \in \mathcal{S}^*$ such that

$$\omega(Y \cap (U^* \setminus C)) \geq \frac{1}{k} \cdot \omega(U^* \setminus C) \geq \frac{1}{k} \cdot (\omega(U^*) - \omega(C)) = \frac{g_t}{k},$$

where $C = \bigcup_{i=1}^{t} X_i$. As X_{t+1} chosen so as to maximize $\omega(X_{t+1} \setminus C)$, we have

$$g_t - g_{t+1} = \omega(X_{t+1} \setminus C) \geq \omega(Y \setminus C) = \omega(Y \cap (U^* \setminus C)) \geq \frac{g_t}{k},$$

and

$$g_{t+1} \le (1 - \frac{1}{k}) \cdot g_t \le (1 - \frac{1}{k})^{t+1} \cdot g_0.$$

Here, notice that $g_0 = \omega(U^*)$.

The analysis of Greedy Procedure for MAX COVERAGE. The output set cover $\{X_1, \ldots, X_k\}$ covers a set of elements of weight sum $g_0 - g_k$, which is at least

$$g_0 - (1 - \frac{1}{k})^k \cdot g_0 = (1 - (1 - \frac{1}{k})^k) \cdot \omega(U^*) \ge (1 - \frac{1}{e}) \cdot \omega(U^*).$$

Therefore, the greedy algorithm is a $(1 - e^{-1})$-approximation for MAX COVERAGE.

1.2 The case of WEIGHTED SET COVER

WEIGHTED SET COVER
Instance: a universe U, a collection $\mathcal{S}$ of subsets of U and a cost function $c : \mathcal{S} \to Q^+$.
Goal: Find a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ of minimum total cost such that $\bigcup_{S \in \mathcal{S}'} S = U$.

We use a similar greedy procedure that we used for SET COVER. That is, we choose a set S so that the ratio of "the number of newly covered elements" and the cost of the set is maximized. For unweighted SET COVER, the cost was 1 per set. For WEIGHTED SET COVER, the cost will be $c(S)$. For the purpose of the approximation ratio analysis, it turns out that using the reciprocal of this ratio is useful.

Algorithm 2 Algorithm for WEIGHTED SET COVER

```
1: procedure SetCover($U, \mathcal{S}, k$)
2:     Mark every element of $U$ as uncovered.
3:     $\mathcal{S}' \leftarrow \emptyset$ and $C \leftarrow \bigcup_{S \in \mathcal{S}'} S$
4:     while $U$ has an uncovered element do
5:         Select $X \in \mathcal{S} \setminus \mathcal{S}'$ which minimizes $c(X)/|X \setminus C|$.
6:         price($e$) $\leftarrow c(X)/|X \setminus C|$ for every element $e \in X \setminus C$.
7:         $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{X\}$
8:         Mark every element of $X$ covered; $C \leftarrow C \cup X$.
9:     return $\mathcal{S}'$.
```

Consider a sequence $X_1, \ldots, X_\ell$ of sets in $\mathcal{S}$ such that each set X_i covers at least one new element, i.e. $\tilde{X}_i := X_i \setminus \bigcup_{j<i} X_j$ is non-empty. Let us define the *price* of an element e covered in this sequence as

$$\mathsf{price}(e) := \frac{c(X_i)}{|\tilde{X}_i|}$$

if e is covered first time by X_i. Then the total cost of these sets equals

$$\begin{aligned}\sum_{i=1}^{\ell} c(X_i) &= |\tilde{X}_i| \cdot \frac{c(X_i)}{|\tilde{X}_i|} \\ &= |\tilde{X}_i| \cdot \mathsf{price}(\text{an element of } X_i) \\ &= \sum_{\substack{e \text{ is covered} \\ \text{by some set } X_i}} \mathsf{price}(e)\end{aligned}$$

The analysis of Greedy Procedure for MAX COVERAGE.

Let $X_1, \ldots, X_\ell$ be the sequence produced by the greedy procedure, and consider a total ordering

$$e_1 \prec e_2, \prec \cdots \prec e_n$$

of U such that $\tilde{X}_i \prec \tilde{X}_j$ whenever $i < j$. In other words, $\prec$ follows the ordering of the time when an element is first covered by some set in the sequence, where ties are broken arbitrarily. Let $\mathsf{price}(e)$ to be the price by the time The key fact for analysis is the following.

Lemma 1. *When the greedy procedure terminates, it holds that*

$$\mathsf{price}(e_k) \leq \frac{c(\mathcal{S}^*)}{n-k+1},$$

where $\mathcal{S}^*$ is the cost of an optimal set cover.

Proof: Let i be the smallest index such that $\{e_i, \ldots, e_k\}$ are all freshly covered by the set Z chosen in line 5. Note that the subset $\{e_i, e_{i+1}, \ldots, e_n\}$ is covered by the collection $\mathcal{S}^*$. That is, there is a sequence $S_1, \ldots, S_p$ of sets in a sub-collection $\mathcal{S}^{**}$ of $\mathcal{S}^*$ which minimally covers $\{e_i, e_{i+1}, \ldots, e_n\}$. Because

$$c(S^{**}) = \sum_{i=1}^{p} c(S_i) = \sum_{i=1}^{p} |\tilde{S}_i| \cdot \frac{c(S_i)}{|\tilde{S}_i|}$$

where $|\tilde{S}_j| = |S_j \cap \{e_i, \ldots, e_n\} \setminus \bigcup_{i<j} S_i|$. By Pigeonhole principle, there exists a set $S_i \in \mathcal{S}^{**}$ with $c(S_i)/|\tilde{S}_i| \leq c(S^{**})/(n-i+1)$. Indeed if not,

$$\sum_{i=1}^{p} |\tilde{S}_i| \cdot \frac{c(S_i)}{|\tilde{S}_i|} > \sum_{i=1}^{p} |\tilde{S}_i| \cdot \frac{c(S^{**})}{n-(i-1)} = \frac{c(S^{**})}{n-i+1} \cdot |\bigcup_{i=1}^{p} \tilde{S}_i| = \frac{c(S^{**})}{n-i+1} \cdot |\{e_i, \ldots, e_n\}| = c(S^{**}),$$

a contradiction.

By the way we choose Z in line 5, we derive

$$\frac{c(Z)}{|Z \setminus C|} = \frac{c(Z)}{|Z \cap \{e_i, \ldots, e_n\}|} \leq \frac{c(S_i)}{|\tilde{S}_i|} \leq \frac{c(S^{**})}{n-i+1} \leq \frac{c(S^{**})}{n-k+1}.$$

□

With Lemma 1, we can rewrite the total cost of the set sequence $X_1, \ldots, X_\ell$ produced by the greedy algorithm as

$$\sum_{i=1}^{\ell} c(X_i) = \sum_{i=1}^{n} \mathsf{price}(e_i) \leq \sum_{i=1}^{n} \frac{c(S^{**})}{n-k+1} \leq (\ln n + c) \cdot c(S^{**})$$

for a small constant $c \leq 1$. That is, the produced set cover is $2 \ln n$-approximation of an optimal set cover.

2 Layering Technique

2.1 2-Approximation for WEIGHTED VERTEX COVER

There are multiple 2-approximation algorithm for WEIGHTED VERTEX COVER, and we see one of them using what's called a layering technique.

WEIGHTED VERTEX COVER
Instance: a graph $G = (V, E)$ with a weight function $\omega : V \to Q^+$.
Goal: Find a vertex cover of G with the minimum weight.

The essential idea of the layering technique is to decompose the weight function into layers of simple weight function so that when you consider the input with each layer of weight function, there is an easy approximation algorithm. For WEIGHTED VERTEX COVER, a 'simple' weight function is a *degree-proportional* one. We start with the following observation.

Lemma 2. *For any vertex cover X of G, it holds that $\sum_{v\in X} deg_G(v) \geq |E|$. In particular, $\omega(X) \leq 2 \cdot \omega(\texttt{opt})$ for any optimal vertex cover* `opt` *whenever the weight function $\omega : V(G) \to Q^+$ satisfies $\omega(v) = c \cdot deg(v)$ for all $v \in V(G)$ for some constant $c > 0$.*

Proof: As every edge e of E is incident with a vertex of X, e contributes at least one to the degree sum $\sum_{v\in X} deg_G(v)$ and the inequality follows. □

Algorithm 3 Algorithm for WEIGHTED VERTEX COVER

```
1: procedure wVC($G, \omega$)
2:     if $G$ is an edgeless graph then return $\emptyset$.
3:     $D \leftarrow$ vertices of degree 0.
4:     $c \leftarrow \min_{v\in V(G)\setminus D}\{\omega(v)/deg(v)\}$.
5:     $\omega'(v) \leftarrow \omega(v) - c \cdot deg(v)$.
6:     $S \leftarrow \{v \in V(G) : \omega'(v) = 0\}$.
7:     return $S \cup$ wVC($G - S - D, \omega'$).
```

Let us analyze the approximation ratio of the procedure **wVC** on the input (G, ω). Because G_{i+1} has strictly smaller number of vertices than G_i, the algorithm ends up in an empty graph or an edgeless graph after at most n iterations.

Let $G_0 := G$ for all $i \geq 0$,

- let $G_i = (V_i, E_i)$ be the graph which form the input to the i-th recursive call of **wVC** at line 7,
- let ω_i be the weight function $\omega_i(v) = c_i \cdot deg_{G_i}(v)$ where c_i the constant fixed at line 4 during the execution of the call **wVC**(G_i, ω_i'),
- let S_i be the vertex subset of G_i identified at line 6 during the execution of the call **wVC**(G_i, ω_i'), and
- k is the index of the last recursive call; G_k edgeless.

Although we defined ω_k, let us set $\omega_k(v) = 0$ for every vertex $v \in V(G_k)$ for notational convenience. As G_k is edgeless, ω_k is also a degree-proportional weight function on G_k like other weight functions ω_i for $i < k$.

What is the final output of the procedure $\mathbf{wVC}$ on (G, ω)? The algorithm will unionize all sets S_i at line 7, so the output is $S := \bigcup_{i=0}^{k} S_i$. Hereinafter, we prove that S is indeed a vertex cover of G and $\omega(S) \leq 2 \cdot \omega(S^*)$, where S^* is an optimal vertex cover of G.

• S **is a vertex cover:** Notice that $V = S \cup \bigcup_{i=0}^{k} D_i$. Therefore, if S is not a vertex cover and $e = uv$ is an uncovered edge of G, then there exist $i \leq j \leq k$ such that $u \in D_i$ and $v \in D_j$. Note that in this case v is present in G_i and u has a neighbor, so should not have been deleted at line 3.

• $\omega(S) \leq 2 \cdot \omega(S^*)$**:**

- Let S^* be an optimal vertex cover of (G, ω). Note that $S^* = \mathtt{opt}'_0$.
- Let $\mathtt{opt}_i$ be an optimal vertex cover of (G_i, ω_i).

We will use the fact for any vertex cover C of G and a vertex subset $Z \subseteq V(G)$, $C \cap Z$ is a vertex cover of $G[Z]$ (why is this true?).

For any vertex $v \in V_0 = V(G)$, we observe

$$(1) \qquad \omega(v) = \sum_{i=0}^{p} \omega_i(v),$$

where p is the largest index such that v appears in G_p. Therefore,

$$(2) \qquad \omega(S) = \omega_0(S \cap V_0) + \omega_1(S \cap V_1) + \cdots + \omega_k(S \cap V_k).$$

Due to Lemma 2 and the fact that both $S \cap V_i$ and $S^* \cap V_i$ are vertex covers of G_i, the following holds.

$$\omega_i(S \cap V_i) \leq 2 \cdot \omega_i(\mathtt{opt}_i) \leq 2 \cdot \omega_i(S^* \cap V_i).$$

From the inequalities (1) and (2),

$$\begin{aligned} \omega(S) &\leq 2 \cdot \omega_0(S^* \cap V_0) + 2 \cdot \omega_1(S^* \cap V_1) + \cdots + 2 \cdot \omega_k(S^* \cap V_k) \\ &= 2 \cdot \omega(S^*) \end{aligned}$$

as claimed.

2.2 3-Approximation for Weighted Feedback Vertex Set

Recall that for Weighted Vertex Cover, the layering technique proposes the following strategy for approximation algorithm: (i) we devise an appropriate 'simple' weight function (degree-proportional for Weighted Vertex Cover), (ii) prove that a greedy solution (such as any vertex cover in Lemma 2) is a good approximation w.r.t the proposed weight function, and (iii) design a polynomial-time procedure which decomposes an arbitrary weight function into simple ones and produces a feasible solution in a greedy manner. We follow the same template to design an approximation algorithm for Weighted Feedback Vertex Set, although the analysis of approximation ratio is slightly more involved.

Lemma 3. *For any minimal feedback vertex set X of a graph G with minimum degree at least 2, the following holds.*

1. $|X| \leq m - n + 1$.

2. $m - n + 1 \leq \sum_{v \in X}(deg(v) - 1) \leq 2(m - n) + |X|$.

Proof:

♦ $|X| \leq m - n + 1$: Notice that there exist $m - n + 1$ edges W incident with X such that $G - W$ is a maximal spanning forest T of G. As for any $e = uv \in W$, at least one of u and v is in X. Orient the edges of W arbitrarily so that the head of each oriented edge is in X. Then every vertex of X must be the head of at least one oriented edge. Indeed, if $v \in X$ is not the head of any oriented edge, then $X \setminus v$ covers all the edges of W. In particular, deleting $X \setminus v$ from G deletes all edges of W (and possibly more) and the remaining set of edges is a subset of T, a forest. This contradicts the minimality of X. Therefore, every vertex of X is the head of at least one oriented edge, implying that there is an injection from W to X. We conclude $|X| \leq |W| \leq m - n + 1$.

♦ $m - n + 1 \leq \sum_{v \in X}(deg(v) - 1)$**:** Let Z be the set of edges incident with X and note that $|Z| \leq \sum_{v \in X} deg(v)$. As $G - X$ is a forest on $V(G) \setminus X$, one can add (at least) $|X|$ edges of Z to the forest $G - X$ to obtain a (not necessarily maximal) spanning forest T. This means that the edge set of G can be partitioned into $|Z| - |X|$ edges incident with X together with $|E(T)|$ edges in T. From $|E(T)| \leq n - 1$, it follows

$$\sum_{v \in X} deg(v) \geq |Z| = m + |X| - |E(T)| \geq m - (n - 1) + |X|,$$

from which the inequality follows.

♦ $\sum_{v \in X}(deg(v) - 1) \leq 2(m - n) + |X|$**:** To see this, we use the equation

$$\sum_{v \in X}(deg(v) - 1) = 2m - \sum_{v \in V \setminus X} deg(v) - |X|$$

and have a good lower bound for the value of $\sum_{v \in V \setminus X} deg(v)$. To the value of $\sum_{v \in V \setminus X} deg(v)$, every edge in the forest $G - X$ contributes 2 and every edge crossing between X and $V \setminus X$ contributes exactly one; the latter type of edges are referred to as crossing edges in the rest of the proof. The value contributed by the forest edges of G is $2 \cdot (n - |X| - \#\text{connected components of } G - X)$ (recall that number of edges in an n-vertex forest with exactly p components is $n - p$).

To have a good lower bound on the number of crossing edges, we use the assumption that the min degree of G is at least 2. Under this assumption, the key observation is that each connected component of the forest $G - X$ is incident with at least two crossing edges. Indeed, a component (tree) T of $G - X$ has one crossing edge incident with its vertex set, then as any tree has at least two leaves, at least one leaf of T has degree 1 in G. This contradicts the min degree assumption. To sum up, we have $\sum_{v \in V \setminus X} deg(v) \geq 2 \cdot (n - |X| - p) + 2p$, where p is the number of connected components of $G - X$ and

$$\begin{aligned}\sum_{v \in X}(deg(v) - 1) &= 2m - \sum_{v \in V \setminus X} deg(v) - |X| \\ &\leq 2m - 2 \cdot (n - |X|) - |X| \\ &= 2(m - n) + |X|.\end{aligned}$$

□

Corollary 1. *Let G be a graph and $\omega : V(G) \to Q^+$ be a weight function which satisfies $\omega(v) = c \cdot deg(v)$ for all $v \in V(G)$ for some constant $c > 0$. Then any minimal feedback vertex set X is a 3-approximation of (G, ω), i.e. $\omega(X) \leq 3 \cdot \omega(\mathtt{opt})$, where $\mathtt{opt}$ is an optimal feedback vertex set of (G, ω).*

Proof: Observe

$$\begin{aligned}
\omega(X) &= \sum_{v \in X} c \cdot (deg(v) - 1) \\
&= c \cdot (2m - 2n + |X|) && \because \text{ 2 of Lemma 3} \\
&= c \cdot (2m - 2n + m - n + 1) && \because \text{ 1 of Lemma 3} \\
&\leq 3c \cdot (m - n + 1) \\
&\leq 3c \cdot \sum_{v \in \mathtt{opt}} (deg(v) - 1) && \because \text{ 2 of Lemma 3} \\
&= 3 \cdot \omega(\mathtt{opt}).
\end{aligned}$$

□

Algorithm 4 Algorithm for WEIGHTED FEEDBACK VERTEX SET

```
1: procedure wFVS($G, \omega$)
2:     Repeatedly delete vertices of degree at most 1.
3:     if $G$ is an empty graph then return $\emptyset$.
                                  ▷ Now every vertex satisfies $deg(v) - 1 \geq 1$.
4:     $c \leftarrow \min_{v \in V(G)}\{\omega(v)/(deg(v) - 1)\}$.
5:     $\omega'(v) \leftarrow \omega(v) - c \cdot (deg(v) - 1)$.
6:     $S \leftarrow \{v \in V(G) : \omega'(v) = 0\}$.
7:     Find a minimal fvs $S^o$ of $G$ contained in $S \cup$ wFVS($G - S, \omega'$).
8:     return $S^o$
```

Induction step. We analyze a single execution of the procedure **wFVS**. Let G be the input graph at the beginning of the procedure, i.e. *before* deleting vertices at line 2. Let D be the deleted vertices at line 2. The following statement is the key to inductively prove the approximation ratio.

Lemma 4. *Assume that S' returned by* **wFVS**$(G' = G - S - D, \omega')$ *is a feedback vertex set of G' with $\omega'(S') \leq 3 \cdot \omega'(\mathtt{opt}')$, where $\mathtt{opt}'$ is an optimal feedback vertex set of (G', ω'). Then*

1. *there exists a feedback vertex set of G contained in $S \cup S'$, and in particular the output S^o at line 7 is indeed a minimal feedback vertex set of G, and*
2. *$\omega(S^o) \leq 3 \cdot \omega(\mathtt{opt})$, where $\mathtt{opt}$ is an optimal feedback vertex set of (G, ω).*

Proof: To show the first statement, it suffices to show that $S \cup S'$ is a feedback vertex set of G. If not, then there is a cycle C in $G - (S \cup S')$. As no vertex of D participates in a cycle of G, C must be fully contained in $G' - S'$. This contradicts the assumption that S' is a feedback vertex set of G'.

To prove the second statement, let $\mathtt{opt}$ be an optimal feedback vertex set of (G, ω) and let $\omega^o(v) := c \cdot (deg_G(v) - 1)$ for all vertices of $G - D$. Notice $\omega(v) = \omega^o(v) + \omega'(v)$ holds for all vertices $v \in V(G) \setminus D$. Moreover, no vertex of D participates in a cycle of G and thus, no vertex of D is contained in a minimal feedback vertex set of G. Now

$$\begin{aligned}
\omega(S^o) &= \omega^o(S^o) + \omega'(S^o) && \because \omega = \omega^o + \omega' \text{ on } S^o \subseteq V(G) \setminus D \\
&= \omega^o(S^o) + \omega'(S^o \cap S) + \omega'(S^o \setminus S) && \\
&= \omega^o(S^o) + \omega'(S^o \setminus S) && \because \omega^o(S^o \cap S) = 0 \text{ due to lines 5 and 6} \\
&\leq 3 \cdot \omega^o(\texttt{opt}) + \omega'(S') && \because \text{Corollary 1 and } S^o \setminus S \subseteq S' \\
&\leq 3 \cdot \omega^o(\texttt{opt}) + 3 \cdot \omega'(\texttt{opt}') && \because \text{Induction hypothesis} \\
&\leq 3 \cdot \omega^o(\texttt{opt}) + 3 \cdot \omega'(\texttt{opt} \cap V(G')) && \because \texttt{opt} \cap V(G') \text{ is a fvs of } G' \\
&\leq 3 \cdot \omega^o(\texttt{opt}) + 3 \cdot \omega'(\texttt{opt}) && \because \texttt{opt} \cap D = \emptyset \\
&= 3 \cdot \omega(\texttt{opt}).
\end{aligned}$$

□

Completing the analysis of wFVS. When G is an empty graph at line 3, it means that before executing line 2 the input graph is acyclic. So the output $\emptyset$ is a valid feedback vertex set. Therefore, the output of $\mathbf{wFVS}(G, \omega)$ is trivially a feedback vertex set of G within triple of an optimal solution value.

Now we apply Lemma 4 inductively on the depth of the recursive calls, with the induction hypothesis that the precondition (S' returned by ...) of Lemma 4 is satisfied. By Lemma 4, the output of the current call $\mathbf{wFVS}(G, \omega)$ is a 3-approximation of an optimal solution. This completes the proof.

3 Remarks

The textbook and lecture note below make a great presentation on approximation algorithms.

- The design of Approximation Algorithms, David P. Williamson, David B. Shmoys, Cambridge University Press. Can be downloaded at https://www.designofapproxalgs.com/.
- Chandra Chekuri's lecture note on Approximation Algorithms is available here.