Submodular set cover or just Submodular cover (generalization of set cover)

Let U be a finite set and let $f: U \to Z^+$ be an integer-valued non-negative function.

Function f is non-decreasing if $f(S) \leq f(T)$ for all $S \subseteq T \subseteq U$

Function f is submodular if $f(S)+f(T) \geq f(S \cup T)+f(S \cap T)$ for all $S, T \subseteq U$

or equivalently if the marginal profit of each item should be non-decreasing

i.e. $f(A \cup \{a\}) - f(A) \leq f(B \cup \{a\}) - f(B)$ if $B \subseteq A \subseteq U$ and $a \in U/B$ $(U-B)$

Function f is subadditive if $f(S)+f(T) \geq f(S \cup T)$

Function f is polymatroid if f is a non-decreasing, submodular (and integer-valued) with $f(\emptyset)=0$.

The submodular cover problem: A finite set $U=\{u_1, \ldots, u_n\}$ and a non-decreasing submodular function f. There is a non negative cost c_i associated with each element $u_i \in U$. The cost of a subset $S \subseteq U$ is $c(S) = \sum_{u_i \in S} c_i$.

A subset $S \subseteq U$ is spanning if $f(S) = f(U)$. The problem asks for a spanning set S^* such that $c(S^*)$ is minimum.

Consider a greedy algorithm which starts from empty set and gradually add elements until we have a spanning set. The element j is picked in each iteration C is locally optimal, i.e., the one maximizing $\frac{c_j}{f(C \cup \{j\}) - f(C)}$.

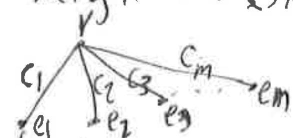
Thm (Wolsey 1982): When f is an (integer-valued) submodular with $f(\emptyset)=0$, the cost of a solution computed by the greedy algorithm above is at most $H(\max_{j \in U} f(\{j\})) = O(\log(\max_{j \in U} f(\{j\})))$ times optimum.

The Proof goes along the same line as in the one for set cover and hence omitted.

Note that set cover is a special case of submodular cover in which each set in the set cover is an "element" of submodular cover and $f(X)$ is the number of elements covered by sets $x \in X$. In this case $f(\{j\})$ is at most n and thus the above algorithm gives $H(n) = O(\log n)$ approximation algorithm for the set cover.

submodular cover has lots of applications, e.g. Capacitated set cover in which each subset has a capacity of the number elements that it "cover" "really" for its cover. You may see more examples in assignments.

Finally you may consider submodular cover as a tree of height one (star) rooted at r such that there is a cost c_i on its ith leaf and



the problem asks for a minimum cost subtree such that its leaves have coverage of the universe U. In general, if U is an arbitrary tree and f is a non-decreasing submodular function with $f(\emptyset)=0$, i.e., f is a polymatroid, the problem is called the submodular tree cover or polymatroid steiner tree.

Thm (Calinescu & Zelikovsky '04): There is a polynomial-time $O((\log n)^{1+\varepsilon} \log k)$-approximation algorithm for polymatroid steiner trees on trees with n nodes and $k = \max_{j \in U} f(\{j\})$.

The Proof is again by a greedy algorithm which is much more involved. ☐

Note that we cannot get any approximation better than $O(\log^{2-\varepsilon} n)$ in this case since polymatroid steiner tree generalizes group steiner tree which has such a lower bound.

→ Also note that in all above algorithms which are polynomial we assume that there is a oracle which gives $f(S)$ for any $S \subseteq U$ in polynomial-time.

Applications in covering by base stations

Maximum coverage: A collection of sets $S=\{S_1, S_2, \ldots, S_m\}$ with cost $cost(S_1), \ldots, cost(S_m)$ over a universe $U=\{x_1, \ldots x_n\}$ and weights $w_1, \ldots w_n$.

Goal: find a collection of sets $S' \subseteq S$, such that the total cost of elements in S' does not exceed L (the budget) and the total weight of elements covered by S' is maximized.

In the unit cost version, the cost of all sets are one.

NP-hardness comes from Set-cover trivially even in the unit cost version of the problem: try the minimum L which can cover everything

Thm: There is a $(1-\frac{1}{e})$ approximation for the budgeted maximum coverge.

In the unit cost case: simple greedy algorithm that picks at each step a set maximizing the weight of the uncovered elements is $(1-\frac{1}{e})$ approx

consider OPT and let $S_1, S_2, \ldots, S_L$ be the sets ~~from opt~~ which are added to our solution Alg at iterations $1 \leq 2 \leq 3 \leq \cdots \leq L$.

Let $G_i = \bigcup_{h=1}^{i} S_h$ and w(a set F) = sum of the weights of elements in F

Then

Lm 1 $w(G_i) - w(G_{i-1}) \geq \frac{1}{L}\left(w(opt) - w(G_{i-1})\right)$ for $i \geq 1$

let w'_i be the total weight of elements in S_i not covered in G_{i-1}. since we are choosing a set S_i with maximum w'_i, and the number of sets in opt is at most L, then

$w(opt) - w(G_{i-1}) \leq L\, w'_i = L\left(w(G_i) - w(G_{i-1})\right)$ and thus we are done ✓

<u>Lm 2</u>: $w(G_i) \geq \left(1 - \left(1 - \frac{1}{l}\right)^i\right) w(opt)$.

Use induction: $w(G_1) \geq \frac{w(opt)}{l}$ ✓

suppose it is correct for $1, \ldots, i-1$, we prove it for i

$$w(G_i) = w(G_{i-1}) + [w(G_i) - w(G_{i-1})]$$

$$\geq w(G_{i-1}) + \frac{1}{l}[w(opt) - w(G_{i-1})] \quad \text{by Lm 1}$$

$$= \left(1 - \frac{1}{l}\right) w(G_{i-1}) + \frac{1}{l} w(opt)$$

$$\geq \left(1 - \frac{1}{l}\right)\left(1 - \left(1 - \frac{1}{l}\right)^{i-1}\right) w(opt) + \frac{1}{l} w(opt) \quad \text{by induction}$$

$$= \left(1 - \left(1 - \frac{1}{l}\right)^i\right) w(opt).$$

Thus $w(G_q) \geq \left(1 - \left(1 - \frac{1}{l}\right)^l\right) w(opt) = \left(1 - \left(1 - \frac{1}{l}\right)^l\right) w(opt) \geq \left(1 - \frac{1}{e}\right) w(opt)$

This does not work when sets have different cost!!!

adding the most cost efficient set

why? x_1 x_2 $S_1 = \{x_1\}$ $S_2 = \{x_2\}$ $L = p+1$

weight 1 p cost 1 $p+1$

W_i: the total weight of elements covered by set S_i. $W_{max} = \max_i \{W_i\}$

opt S_2 with weight p, $ALG = S_1$ with weigh 1.

Thus the greedy does not work: if we output the best of greedy and the set which covers the maximum weight W_{max} then $\frac{1}{2}\left(1 - \frac{1}{e}\right)$ approximation (essentially we can say if we could add the last set in opt to ALG, we are good and we know its weight $\leq W_{max}$)

Anyway, we can improve this also to $\left(1 - \frac{1}{e}\right)$ for general cost (try all subsets of size k at the beginning) $\boxed{k \text{ is fixed}}$

★ A simple reduction from the hardness $(1-\varepsilon)\ln n$ for set cover shows that maximum coverage (even with unit costs) is not approximable within $\left(1 - \frac{1}{e} - \varepsilon\right)$ unless $NP \subseteq DTIME(n^{\log\log n})$

(see the next page)

Hardness Thm: Even the unit cost version of the maximum coverage problem cannot be approximated within a factor better than $(1-\frac{1}{e})$, unless $NP \subseteq DTIME(n^{\log\log n})$

First we need this theorem of Feige

Feige Thm: If a set cover problem is approximable within a factor of $(1-\epsilon)\ln n$ for any $\epsilon > 0$ then $NP \subseteq DTIME(n^{\log\log n})$

- Consider unit cost set cover. Assign unit weight to each element.
- suppose there is an algorithm A with approximation factor $\alpha > (1-\frac{1}{e})$
- guess the number of sets in set-cover, say k.

Thus the number of elements covered by k sets is at least n.

New Algorithm setALG for set-cover

- Run A, it covers αn, were $\alpha > (1-\frac{1}{e})$ just for analysis, choose sets for cover C
- Remove C and elements covered and reiterate.

let the number of uncovered elements at the start of the iteration i is n_i

setALG covers αn_i with k sets. $n_{i+1} \leq n_i(1-\alpha)$

Say we iterate $l+1$ times then $n_l \geq 1$

$1 \leq n_l \leq n_{l-1}(1-\alpha) \leq n_{l-2}(1-\alpha)^2 \leq n(1-\alpha)^l$ thus $l \leq \frac{\ln n}{\ln(\frac{1}{1-\alpha})} + 1$

thus our set cover has size $kl \leq k\frac{\ln n}{\ln(\frac{1}{1-\alpha})} = OPT \frac{\ln n}{\ln(\frac{1}{1-\alpha})}$

Since $\alpha > (1-\frac{1}{e})$, $\ln(\frac{1}{1-\alpha}) > 1$ which contradicts Feige's Thm above. ✓

$\frac{1}{1-\alpha} > e$