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

Let $U$ be a finite set and let $f: U \rightarrow \mathbb{Z}^+$ 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 $a \notin B \cup (U \setminus B)$.

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

Function $f$ is **poly-matroid** 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 is locally optimal, i.e., the one maximizing $c_j f(C \cup \{j\}) - f(C)$.

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 $X \in \mathcal{X}$. In this case $f(\{j\})$ is at most $\frac{n}{m}$ and thus the above algorithm gives $H(n) = \Theta(\log n)$ 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 "covers" really for its cost. You may see more examples in assignments.

Finally, you may consider submodular cover as a tree of height one (star) rooted at the leaf such that there is a cost $c_i$ on its $i$th 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 of polymatroid Steiner tree.

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

The proof is again by a greedy algorithm which is much more involved. Note that we cannot get any approximation better than $O(\log n)$ 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 an oracle which gives $f(S)$ for any set $S \subseteq U$ in polynomial-time.
Applications in covering by base stations

Maximum coverage: A collection of sets $S = \{S_1, S_2, \ldots, S_m\}$ with costs $\text{cost}(S_1), \ldots, \text{cost}(S_m)$ over a universe $U = \{x_1, x_2, \ldots, x_n\}$ 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 coverage.

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 $\text{OPT}$ and let $S_1, S_2, \ldots, S_L$ be the sets which are added to our solution alg at iterations $1 \leq i \leq 2 \leq \cdots \leq L$.

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

Then
\[
\lim_{i \to \infty} \frac{w(G_i) - w(G_{i-1})}{i} \geq \frac{1}{L} (w(\text{OPT}) - w(G_{i-1})) \quad \text{for } i \geq 1.
\]

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 $\text{OPT}$ is at most $L$, then
\[
w(\text{OPT}) - w(G_{i-1}) \leq L \ w_i = L (w(G_i) - w(G_{i-1}))
\]

and thus we are done.
Lm2: \( w(G_i) \geq (1 - (1 - \frac{1}{L})^i) w(\text{opt}) \).

Use induction: \( w(G_i) \geq \frac{w(\text{opt})}{L} \)

Suppose it is correct for \( i \leq k-1 \), we prove it for \( i \).

\[
\begin{align*}
w(G_i) &= w(G_{i-1}) + [w(G_i) - w(G_{i-1})] \\
&\geq w(G_{i-1}) + \frac{1}{L} [w(\text{opt}) - w(G_{i-1})] \quad \text{by Lm 1} \\
&= (1 - \frac{1}{L}) w(G_{i-1}) + \frac{1}{L} w(\text{opt}) \\
&\geq (1 - \frac{1}{L}) \left(1 - (1 - \frac{1}{L})^{i-1} \right) w(\text{opt}) + \frac{1}{L} w(\text{opt}) \quad \text{by induction}
\end{align*}
\]

\[
(1 - (1 - \frac{1}{L})^i) w(\text{opt}) \geq (1 - \frac{1}{L})^i w(\text{opt})
\]

Thus \( w(G_k) \geq (1 - (1 - \frac{1}{L})^k) w(\text{opt}) \geq (1 - (1 - \frac{1}{L})^k) w(\text{opt}) \geq (1 - \frac{1}{L})^k w(\text{opt}) \).

Adding the most cost efficient set

\[
\begin{array}{ccc}
\text{why?} & x_1 & x_2 \\
\text{weight} & 1 & p \\
\text{cost} & 1 & p+1 \\
\end{array}
\]

\( \text{opt } S_2 \) with weight \( p \), \( \text{Alg } = S_1 \) with weight \( 1 \)

Thus the greedy does not work: if we output the best of greedy and the set which covers the maximum weight then \( \frac{1}{2} (1 - \frac{1}{L})^k \) approximation (essentially we can say if we could add the last set in \( \text{opt} \) to \( \text{Alg} \), we are good and we know its weight \( \leq \text{opt} \)).

Anyway, we can improve this also to \( (1 - \frac{1}{L})^k \) for general cost

A simple reduction from the hardness \( (1 - \frac{1}{L})^k \) for set cover shows that maximum cover (even with unit costs) is not approximable within \( (1 - \frac{1}{L})^k \) (see next page)
Hardness Theorem: Even the unit cost version of the maximum coverage problem cannot be approximated within a factor better than \((1-\frac{1}{e})\), unless \(NP \subseteq \text{DTIME}(n^{O(\log n)})\).

First we need this theorem of Feige:

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

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 for set-cover

- Run \(A\), it covers \(an\) and have \(\alpha > (1-\frac{1}{e})\), choose sets to cover C.
- Remove C and elements covered and reiterate.

Let the number of uncovered elements at the start of the iteration \(i\) is \(n_i\).

Set \(\frac{1}{\alpha} \geq \frac{n_i}{n}\) for \(i\) such that \(n_i \leq \ln n\).

Thus \(i \leq \ln n + 1\).

Thus our set cover has size \(K \leq k \ln n = \text{OPT} \ln n\).

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