We'll select the sites and the users they cover using the idea of the Set-Cover greedy algorithm. If a site $s$ is used to cover the a subset $U_s$ of users, then the average user cost is $(f_s + \sum_{u \in U_s} d_{us})/|U_s|$. The idea behind the greedy algorithm is to select the site $s$ with a subset $U_s$ that minimizes this quantity. First we need to argue that this minimum can be found.

(1) Given a set $R$ of uncovered users, and a site possible $s$, one can find the subset $U_s \subset U$ that minimizes the average cost $(f_s + \sum_{u \in U_s} d_{us})/|U_s|$ in polynomial time.

Proof. Sort the users by increasing distance $d_{us}$ from site $s$. The set $U_s$ will be an initial set of this sorted sequence: $U_s = \{ u \in R : d_{us} \leq \alpha \}$ for some value $\alpha$. ■

Now the algorithm will be analogous to the Set Cover greedy algorithm. We select sites $s$ with subsets $U_s$ by the above greedy rule: selecting the site and the set that minimizes the average cost of covering a new user. There is one more option to consider. Suppose $T$ is the subset of sites already selected. For a site $s \in T$ we can add a new node $u \in R$ to $U_s$, covering the new user $u$ at the cost of $d_{us}$. In the algorithm given below, we will also save the cost $c_u$ at which user $u$ got covered by the algorithm. These values will be used by the analysis.

Start with $R = U$ and $T = \emptyset$.

While $R$ is not empty

Let $c = \min_{u \in R, s \in T} d_{us}$

Select $s \in S - T$, and set $U_s \subset R$ that minimizes

$c' = (f_s + \sum_{u \in U_s} d_{us})/|U_s|$.

If $c' \leq c$ then

Select the site $s$ and set $U_s$ used to obtain $c'$ above.

Add $s$ to $T$, and delete $U_s$ from $R$.

Set $c_u = c'$ for all $u \in U_s$.

Else

Select $s$ and $u$ obtaining the first minimum.

Add $u$ to $U_s$,

Set $c_u = c$.

Endwhile

First, note that if we select the set of sites $T_s$ and have each site $s \in T_s$ cover the users in $U_s$, then we get a solution to the problem with total cost $\sum_{u \in T_s} c_u$. Also, the algorithm runs in polynomial time. It remains to show that this is an $H(n)$ approximation algorithm.

The proof of the approximation ratio follows very closely the proof for the set cover algorithm. Consider an optimum solution. Assume it contains a subset $T^*$ of sites, and $s \in T^*$ is used to cover a set $U^*_s$ of users. The cost of using $s$ to cover $U^*_s$ is $f_s + \sum_{u \in U^*_s} d_{us}$. We will want to compare the optimum’s cost, and $\sum_{u \in U^*_s} c_u$, which is the cost our greedy algorithm paid for the users in $U^*_s$.

\footnote{ex37.588.671}
(2) Using the notation introduced above, and the costs defined by the algorithm, we have that \( \sum_{u \in U^*_s} c_u \leq H(d)(f_s + \sum_{u \in U^*_s} d_{us}) \), where \( d = |U^*_s| \).

Proof. For notational simplicity, let \( \mathcal{C} = f_s + \sum_{u \in U^*_s} d_{us} \). Consider the elements in \( U^*_s \) in the order the algorithm covered them. Assume they are \( u_1, u_2, \ldots, u_d \). Consider the moment the algorithm covers the \( i \)th node \( u_i \) from \( U^*_s \). There are two cases to consider.

Case 1. At this point of the algorithm \( s \notin T \).

Case 2. At this point of the algorithm \( s \in T \).

When the algorithm covered \( u_i \) it selected the smallest average cost. In Case 1 this implies that the cost \( c_{u_i} \) is at most the cost of selecting \( u_i \) with the set \( U^*_s \cap R \), which is at most \( c_{u_i} \leq \mathcal{C}/(d - i + 1) \) (as \( i - 1 \) previously covered nodes are no longer in the set). In Case 2, this implies that \( c_{u_i} \leq d_{us} \). Assume that Case 1 applies when the first \( k \) nodes are covered, and after that Case 2 applies (\( k \) may be equal to \( d \)). Now summing all costs in \( U^*_s \) we get that

\[
\sum_{u \in U^*_s} c_u \leq \mathcal{C}/d + \mathcal{C}/(d - 1) + \ldots + \mathcal{C}/(d - k + 1) + \sum_{i > d} d_{u_i, s}.
\]

Now if \( d = k \) then the upper bound on the cost is \( H(d)\mathcal{C} \) as claimed. If \( k < d \) then note that the costs \( \sum_{i > d} d_{u_i, s} \) is bounded by \( \mathcal{C} \), and so we also can bound the total cost by \( H(d)\mathcal{C} \). 

Now we are ready to prove that the algorithm is an \( H(n) \) approximation algorithm. Let \( T^* \) and \( U^*_s \) be the optimal solution. The total cost of the solution is \( \sum_{s \in T^*}(f_s + \sum_{u \in U^*_s} d_{us}) \). We use the above Lemma to bound each term of the cost, and upper bound \( H(d) \) by \( H(n) \) for each set \( U^*_s \) in the optimum, to get the following.

\[
OPT - \sum_{s \in T^*}(f_s + \sum_{u \in U^*_s} d_{us}) \leq \sum_{s \in T^*} H(n) \sum_{u \in U^*_s} c_u - H(n) \sum_{s \in U} c_u,
\]

where the last sum is the algorithm’s cost as claimed by the first Lemma.