Hardness of Asymmetric k-center

April 25, 2012

In order to prove the hardness, we will focus on instances of Asymmetric k-center (AkC) of the following form. We have $h+1$ layers $V_0, \dots, V_h$ of vertices. Layer V_0 contains a single vertex s. Layer V_i contains some set of n_i vertices. All edges are between pairs of consecutive layers, directed from V_{i-1} to V_i. There is an edge from s to every vertex of V_1. The cost of the optimal solution is 1.

Every pair V_{i-1}, V_i of consecutive layers can be viewed as an instance of the Set Cover problem, where vertices of V_{i-1} serve as sets, vertices of V_i as elements, and element $u \in V_i$ belongs to set $v \in V_{i-1}$ iff there is an edge (v, u) in the graph. Denote this Set Cover instance by SC_{i-1}. Let k_i denote the cost of the optimal solution to the Set Cover instance SC_i. Consider the optimal solution to the AkC problem instance. This solution must contain s (since this is the only way to cover it), and for every layer V_i, the subset $S_i \subseteq V_i$ of vertices that belong to the solution must define a feasible set cover solution for SC_i. So $|S_i| \geq k_i$, and the total number of centers in this solution is at least $1 + \sum_{i=1}^{h-1} k_i$ and at most k. Therefore, $\sum_{i=1}^{h-1} k_i < k$.

In order to find an h-approximate solution, we need to find k vertices covering all vertices within distance h. Let S be such a solution. Then $s \in S$, since this is the only way to cover s. Since we are allowed a covering radius of h, s covers all vertices in layers $V_1, \dots, V_{h-1}$ within distance h. We can assume w.l.o.g. that all other vertices in S belong to V_1: otherwise, if $v \in S$ with $v \in V_i$ for $i > 1$, then we can replace v with its ancestor in layer V_1, and the solution remains feasible. So finding an h-approximate solution is equivalent to selecting $k-1$ vertices in V_1 that cover all vertices in V_h. This framework is very similar to the approximation algorithm.

We will show a reduction from the SAT problem to this restricted type of AkC problem, with $h = \Omega(\log^* n)$, such that:

- If the input formula φ is satisfiable (we call it a yes-instance), then there is a collection of k vertices covering all vertices within radius 1.
- If φ is not satisfiable (no-instance), then no set of k vertices in V_1 covers all vertices in V_h.

So if we have an h-approximation algorithm for the AkC problem, then this algorithm will distinguish between satisfiable and unsatisfiable SAT formulas. Therefore, AkC is hard to approximate up to factor $h = \Omega(\log^* n)$.

1 The Construction

Given the SAT formula φ, we construct our instance of AkC as above. The specific Set Cover instances that we plug in at each level depend on the formula φ. What we need from these SC instances is:

- If φ is satisfiable: there is a "cheap" solution to each Set Cover instance.
- If φ is not satisfiable, then even if we select almost all the sets, still a significant number of elements is not covered.

We will use the following useful result:

Theorem 1 *Given a SAT formula φ over n variables, and a parameter d, we can construct an instance $SC(\varphi, d)$ of the Set Cover problem with N elements and M sets, such that:*

- *If φ is a yes-instance, then there is a collection of M/d sets covering all elements.*
- *If φ is a no-instance, then any collection containing at most $M(1 - 1/d)$ sets covers at most $N \cdot \left(1 - \frac{1}{2^{d^\beta}}\right)$ elements. (β is a fixed constant, say $\beta = 20$).*
- *$N \leq n^{O(\log d)} 2^{d^\beta}$, $M \leq N \leq 2^{d^\beta} M$, and the running time of the reduction is polynomial in N.*

We now go back to the AkC construction. The basic idea is to plug in $SC(\varphi, d_i)$ instead of SC_i. We need to choose the parameters d_i so that in the no-instance, no choice of k vertices in the first layer will be sufficient to cover all vertices in the last layer. It is enough to choose d_1 to be some large enough constant, say β^2 and the recursive formula is:

$$d_{i+1} = 10 \cdot 2^{d_i^{2\beta}}$$

Notice that for each i, the vertices of V_i serve as the elements for instance SC_{i-1}, and as sets for instance SC_i. We cannot plug the set cover instances in directly, since we need to ensure that the number of sets in SC_i equals to the number of elements in SC_{i-1}. We overcome this difficulty in a straightforward way: by creating the "right" number of copies of each Set Cover instance.

Let N_i denote the number of elements of $SC(\varphi, d_i)$ and M_i denote the number of sets in it. We will create $X_1 = \prod_{i=2}^{h} M_i$ copies of $SC(\varphi, d_1)$, and in general, for all i, we create $X_i = \prod_{j=1}^{i-1} N_j \prod_{j=i+1}^{h} M_j$ copies of $SC(\varphi, d_i)$. We denote by SC_i the resulting Set Cover instance. Then:

- The number of sets in SC_i is $M'_i = \prod_{j=1}^{i-1} N_j \prod_{j=i}^{h} M_j$
- The number of elements in SC_i is $N'_i = \prod_{j=1}^{i} N_j \prod_{j=i+1}^{h} M_j$

So $N'_i = M'_{i+1}$ is the number of vertices in V_{i+1}. Now for each $1 \leq i < h$, we partition the vertices of V_i into X_i subsets of M_i vertices each, and the vertices of V_{i+1} into X_i subsets of N_i vertices each. We then choose X_i disjoint pairs of subsets. Each such pair contains one subset from V_i and one subset from V_{i+1}. For each such pair, we construct a copy of the set cover instance $SC(\varphi, d_i)$, by adding the edges corresponding to $SC(\varphi, d_i)$ to the graph. This completes the construction description.

To finish the hardness of approximation proof, we need three things: define k and analyze the yes-instance; Show that in the no-instance we cannot choose k vertices of V_1 to cover all vertices in V_h; Compute the size of the final graph and compute the hardness of approximation factor we obtain.

2 Yes Instance and the Choice of k

Denote by $k_i = X_i M_i / d_i$ - the cost of the set cover solution for SC_i if φ is a YES-INSTANCE. We then set $k = 1 + \sum_{i=1}^{h} k_i$. Clearly, if φ is a YES-INSTANCE, there is a solution to the k-center problem containing k vertices that cover all other vertices within distance 1. This solution consists of the union of solutions to the set cover instances SC_i, plus the vertex s. We need to bound k.

Claim 1 $\sum_{i=1}^{h} k_i \leq 2k_1$.

Proof: It is enough to prove that for all i, $k_i \leq k_{i-1}/2$. Then we get a geometric series and the result follows. We now prove that $k_i \leq k_{i-1}/2$.

$$
\begin{aligned}
k_i &= \frac{M_i X_i}{d_i} \\
&= \frac{N_{i-1} X_{i-1}}{d_i} && \text{(because } |V_i| = M_i X_i = N_{i-1} X_{i-1}\text{)} \\
&= \frac{M_{i-1} X_{i-1}}{d_{i-1}} \frac{N_{i-1}}{M_{i-1}} \frac{d_{i-1}}{d_i} && \text{(just multiplying and dividing by } d_{i-1} M_{i-1}\text{)} \\
&\leq k_{i-1} \cdot \frac{2^{d_{i-1}^{\beta}} \cdot d_{i-1}}{d_i} && \text{(because } N_{i-1} \leq M_{i-1} \cdot 2^{d_{i-1}^{\beta}} \text{ from Theorem 1)} \\
&\leq \frac{k_{i-1}}{2} && \text{(from definition of } d_i\text{)}
\end{aligned}
$$

□

3 No-Instance Analysis

Let S be any subset of $2k_1$ vertices in the first layer. Our goal is to show that there is at least one vertex in the last layer that is not covered by S. Specifically, we'll show the following:

Claim 2 *For each $i > 1$, at least $3/d_i$-fraction of vertices of V_i are not covered by S.*

Proof: By induction. For $i = 1$, at least a fraction $3/d_1$ of vertices do not belong to S (if $d_1 > 6$).

Let S_i be the set of vertices covered by S in layer i. Recall that V_i is the union of sets of X_i copies of SC_i. A copy C of SC_i is good iff the number of sets of C belonging to S_i is at most $M_i(1 - 1/d_i)$.

Claim 3 *At least $1/d_i$-fraction of copies of SC_i are good.*

Proof: Assume otherwise. Then there are at least $X_i(1 - 1/d_i)$ bad copies, each of which has at least $M_i(1 - 1/d_i)$ vertices covered. So overall the number of vertices of V_i that are covered is at least $M_i X_i (1 - 1/d_i)^2 > M_i X_i (1 - 3/d_i) = |V_i|(1 - 3/d_i)$.

□

Let C be a good copy of SC_i. Then at least $1/2^{d_i^{\beta}}$ elements of C are not covered, from Theorem 1. Overall, at least $X_i N_i / (d_i \cdot 2^{d_i^{\beta}})$ vertices of V_{i+1} are not covered. Since $d_{i+1} = 10 \cdot 2^{d_i^{2\beta}}$, this is more than $3/d_{i+1}$-fraction of vertices of V_{i+1}.

□

4 Setting the Parameters

Let $|V|$ denote the total instance size. The largest layer in the construction is the last layer, so $|V| \leq h|V_h| = h\prod_{i=1}^{h} N_i \leq h\prod_{i=1}^{h} n^{O(\log d_i)} 2^{d_i^\beta} \leq hn^{O(h\log d_h)} 2^{d_h^\beta}$

Clearly, we have to stop before $2^{d_h^\beta}$ becomes super-polynomial. So we need to bound d_h. We will show that for $h = \Theta(\log^* n)$, $d_h \leq \log\log n$. Assuming this is true,

$$|V| \leq O(\log^* n) \cdot n^{O(\log^* n \log^{(3)} n)} \cdot 2^{(\log\log n)^\beta} \leq n^{O(\log\log n)}$$

From here we get that the reduction runs in tome $n^{O(\log\log n)}$, and the hardness factor that we get is $h = \Omega(\log^* n)$. But since $\log n \leq |V|$, we get that $\log^* n - 1 \geq \log^*(|V|)$. So $h = \Omega(\log^*(|V|))$.

To conclude, we have shown a reduction that, given a SAT formula φ of size n constructs an instance of size $N \leq n^{O(\log\log n)}$, in time poly(N), such that:

- If φ is satisfiable, the AkC instance has solution of cost 1.
- If it is not satisfiable, any solution has cost at least $\Omega(\log^* N)$.

Therefore, unless NP has algorithms running in time $n^{O(\log\log n)}$, AkC is $\Omega(\log^* n)$ hard to approximate. Using the same ideas, we can show that for any constant c, there is no c-approximation for AkC unless $P = NP$.

It now remains to show that for some $h = \Theta(\log^* n)$, $d_h \leq \log\log n$. We first prove the following claim.

Claim 4 *For all* $i \geq 1$, $\log^{(2i)} d_i \leq d_1$.

Proof: The proof is by induction. For $i = 1$, $\log\log d_1 \leq d_1$. Consider now some i.

$$\log^{(2i+2)} d_{i+1} = \log^{(2i+2)} 2^{d_i^\beta} = \log^{(2i+1)} d_i^\beta \leq \log^{(2i)} d_i \leq d_1$$

(since $\log d_i^\beta \leq d_i$). □

Therefore, $\log^{(2h)} d_h \leq d_1$. Since d_1 is a constant, for some constant c, $\log^{(2h+c)} d_h \leq 1$. We set $h = \frac{\log^* n - 3 - c}{2}$. Then $2h + c = \log^* n - 3$. Denote $2h + c = z$. We then get that $\log^{(z)} d_h \leq 1$, but $\log^{(z)}(\log\log n) > 1$ (because $z = \log^* n - 3$, this follows from the definition of $\log^* n$). This can only happen if $h \leq \log\log n$.