# Hardness of Approximating Set Cover

Yuval Filmus

January 2011

## 1 Introduction

This talk describes Feige's result on the hardness of approximation of set cover. We will start by following (somewhat anachronistically) the footsteps of Lund and Yannakakis, showing an $\Omega(\log n)$ hardness result. We will improve it, using Feige's ideas, to $(1-\epsilon)\log n$. For technical reasons, the results are not NP-hardness but rely on a slightly stronger hypothesis, namely that NP cannot be solved in time $n^{O(\log\log n)}$.

In this talk, $\log n$ will always be taken to base $e$.

## 2 Set Cover

Set cover is the following problem: Given a universe $U$ and a collection of subsets $S_i \subset U$ which together cover $U$, find the minimal-size collection covering $U$.

Set cover has several simple approximation algorithms, all achieving more-or-less the same worst-case ratio. The simplest one is the greedy algorithm: In each step, add a set covering the maximal number of elements not already covered.

**Lemma 2.1.** *The greedy algorithm gives a* $(1-o(1))\log n$ *approximation, where* $n = |U|$.

*Proof.* Let $c_t$ be the number of elements *not* covered by the greedy algorithm after $t$ sets are picked. Thus $c_0 = n$. Denote by $m$ the size of the minimal set cover. Suppose that among the first $t$ sets picked by the greedy algorithm, $r$ belonged to the minimal set cover. The other $m-r$ sets cover all the uncovered elements, and so the set picked by the greedy algorithm covers at least $c_t/(m-r)$ elements. Therefore

$$c_{t+1} \leq \left(1 - \frac{1}{m-r}\right) c_t \leq \left(1 - \frac{1}{m}\right) c_t.$$

The process stops when $c_t < 1$, which happens after roughly $m \log n$ steps. $\square$

## 3 From 3SAT to Set Cover

Our starting point is the PCP theorem: it is NP-hard to distinguish satisfiable instances of 3SAT from $(1-\epsilon)$-satisfiable instances, for some $\epsilon$. It will be advantageous to recast this as a two-prover game (we've already seen that):

- Prover 1 assigns a code in $\{0, \dots, 7\}$ for each clause; the code signifies which literal is true (it will be advantageous for the prover to never use the code 0).
- Prover 2 assigns a truth value for each variable.

The verifier selects a clause at random, and a random variable within the clause. She probes the truth assignment of the clause from Prover 1, and the truth assignment of the variable from Prover 2. She is "convinced" if the two answers are consistent, i.e. the underlying value of the variable is the same. The PCP theorem can be recast as follows: it is NP-hard to distinguish between 3SAT instances in which the provers can always convince the verifier, and instances in which the verifier is convinced with probability at most $1-\epsilon$ (for a different $\epsilon$).

We will try to model the situation in the setting of set cover. We will have a set $S(q, a, i)$ for every question-answer pair of Prover $i$. When $i = 1$, the question $q$ is a clause, and the answer $a$ is the truth-value of the literals. When $i = 2$, the question $q$ is a variable, and the answer $a$ is its truth-value. Denote by $Q_i$ the number of possible questions to Prover $i$. Thus $Q_1$ is the number of clauses, and $Q_2$ is the number of variables.

A set cover represents a "single-minded" strategy if every question has a single answer. We would like the set cover instance to have the following property: it must contain an answer for every possible question, and a single-minded strategy is a set cover if and only if the corresponding strategy for the provers always convinces the verifier, i.e. the 3SAT instance is satisfiable.

The remaining piece of the puzzle is thus a "verifier". The verifier can ask the provers $V = 3Q_1$ different questions $(q_1, q_2)$, and each of them will be represented by a part $U_{q_1,q_2}$ of the universe $U$. Each of these parts should be coverable by any consistent pair $S(q_1, a_1, 1), S(q_2, a_2, 2)$. Thus we posit the existence of a partition

$$U_{q_1,q_2} = U_{q_1,q_2,\alpha,1} \cup U_{q_1,q_2,\alpha,2}$$

for every possible consistent answer $\alpha$. What is $\alpha$? It is the value of the variable chosen within the clause. Thus there are two possible partitions for each $U_{q_1,q_2}$, corresponding to the two values of the underlying variable.

Finally, let us connect the prover part of the construction to the verifier part by defining the sets $S(q, a, i)$. The set $S(q, a, i)$ signifies that Prover $i$ answers $a$ when confronted by question $q$. When the verifier asks $(q_1, q_2)$, if it gets consistent answers $(a_1, a_2)$ then $U_{q_1,q_2}$ should be covered. So $S(q, a, i)$ is the union of all sets $U_{q_1,q_2,\alpha,i}$ in which $q_i = q$ (i.e. Prover $i$ is actually asked $q$) and $\alpha$ is the value of the marked variable extracted from $a$ (in the case of Prover 2, $\alpha = a$).

Given a satisfiable 3SAT instance, there is a single-minded strategy for the provers, which can be translated to a set cover. The set cover consists of an answer for every possible question, and so its size is $Q_1+Q_2$. In the next section, we find out what happens when the 3SAT instance is not satisfiable.

## 4 From Set Cover to 3SAT

If the 3SAT instance is not satisfiable, then no single-minded strategy translates to a set-cover. In this section we show how to convert a non-single-minded set cover to a randomized strategy for the provers, with the goal of showing that unless the set cover is very large, the implied strategy is too successful in convincing the verifier.

Given a set cover, how would we construct a strategy for the provers? First, note that any set cover must contain at least one answer (across both provers) for every question $(q_1, q_2)$, since otherwise the corresponding part of the universe $U_{q_1,q_2}$ is not covered at all. We can ensure that an answer exists for both provers by having an element $u_{q_1,q_2,i}$ which belongs only to sets of the form $U_{q_1,q_2,\cdot,i}$. This way we enforce the existence of an answer for every possible question targeted at any of the provers.

Because the strategy is not single-minded, there might be more than one possible answer in the set cover. We will adapt the following natural (and somewhat naive) randomized strategy for the provers: when Prover $i$ is confronted with question $q$, she picks a random answer $a$ from all the sets of the form $S(q, a, i)$ participating in the cover. Our goal is to show that this strategy convinces the verifier with reasonable probability; this implies that some *deterministic* strategy achieves the same amount of success, violating the "gap" in the statement of the PCP theorem.

At this point it becomes apparent that the gap between $1 - \epsilon$ and 1 is not enough for our purposes; the strategy we are going to construct is going to succeed with probability much smaller than $1-\epsilon$. Fortunately, parallel repetition (covered in one of the previous talks) comes to our rescue. The verifier is now going to ask the provers $C$ questions in parallel; the provers will answer these questions in parallel. The verifier will perform her consistency check on all $C$ pairs of answers in parallel. If the 3SAT instance is satisfiable, the provers can always convince her. Otherwise, she will be convinced with probability at most $\delta^C$, where $\delta$ is a constant depending only on $\epsilon$.

Generalizing our construction from the original two-prover system to its parallel repetition is straightforward. The number of questions to each prover $Q_1, Q_2$ is simply raised to the power $C$, as is the number of challenges $(q_1, q_2)$ by the verifier. Each part of the universe $U_{q_1,q_2}$ can now be covered by $2^C$ different partitions $U_{q_1,q_2,\alpha,i}$, since $\alpha$ is now a vector of length $C$.

Suppose the verifier asks the questions $(q_1, q_2)$, and the provers answer $(a_1, a_2)$. When do the provers convince the verifier? Each $S(q_i, a_i, i)$ contains a unique set of the form $U_{q_1,q_2,\alpha_i,i}$. The verifier is convinced exactly when

$\alpha_1 = \alpha_2$. We want to argue that unless the set cover is very large, such an event is bound to happen with reasonable probability.

To start with, let $A_i$ be the number of sets in the set cover pertaining to Prover $i$. How many of them do we see on an average query of the verifier? In the case of Prover 1, because all clauses are as likely to be asked about by the verifier, the average query results in $A_1/Q_1$ possible answers. However, the same is not necessarily true for Prover 2, since some variables may appear in more clauses than other variables. However, it is not hard to reduce an arbitrary 3SAT formula so that each variable appears in exactly 5 clauses; moreover, the PCP theorem remains true even for this variant of 3SAT. Given that, the average query results in $A_2/Q_2$ possible answers.

The typical query thus has $A_i/Q_i$ answers to choose from for Prover $i$. We would like to argue that if $A_1/Q_1 + A_2/Q_2$ is "small", then those sets must contain two consistent answers. The only way in which this will *not* happen is if at most one set from each partition $U_{q_1,q_2,\alpha,\cdot}$ is used. Recall that each of the $A_1/Q_1 + A_2/Q_2$ sets contains a unique set of the form $U_{q_1,q_2,\cdot,\cdot}$. How many of them are needed to cover all of $U_{q_1,q_2}$, if we cannot use any of the built-in partitions?

If we put all the $2^C$ partitions in a sequence and choose one set of each in a greedy manner, we will cover $U_{q_1,q_2}$ with at most $\log_2 |U_{q_1,q_2}|$ sets; for brevity, define $m = |U_{q_1,q_2}|$. It turns out that if the partitions are chosen randomly, the greedy algorithm is almost optimal: we need at least $(1-\epsilon_P)\log_2 m$, where $\epsilon_P \to 0$ as $m \to \infty$; such a "design" can also be constructed explicitly.

So as long as $A_1/Q_1 + A_2/Q_2 < \log_2 m$, the set of answers will contain a pair of consistent answers. This pair will actually be chosen by the provers with probability

$$\frac{Q_1}{A_1} \cdot \frac{Q_2}{A_2} > \frac{4}{(\log_2 m)^2};$$

note that the worst possible case is when $Q_1/A_1 = Q_2/A_2$, from which we get the bound.

The careful listener will have noticed that $A_1/Q_1 + A_2/Q_2$ is only the *average* case: it might be that most of the time the sets of answers are larger than $\log_2 m$, and occasionally they are very small. Here Markov's inequality comes to our rescue: with constant probability we reach a set of answers that is at most only slightly larger than the average (I'll let you work out the details).

What does the argument give us? If the 3SAT formula is satisfiable, then there is a set cover of size $Q_1 + Q_2$. Otherwise, if $A_1/Q_1 + A_2/Q_2 < \log_2 m$ (glossing over some epsilons) then the provers have a strategy which convinces the verifier with probability $4/(\log_2 m)^2$. If $4/(\log_2 m)^2 > \delta^C$, then this would contradict the parallel-repetition-amplified PCP gap. We would then deduce that $A_1/Q_1 + A_2/Q_2 \geq \log_2 m$. We want to conclude that $A_1 + A_2$ is big.

Here we hit some trouble: $Q_1$ is much bigger than $Q_2$. Whereas the size of the "good" cover is approximately $Q_1$, the best lower bound we can get on $A_1 + A_2$ is $Q_2 \log_2 m$, which is meaningless! The way to solve this problem is to somehow make $Q_1$ and $Q_2$ coincide This can be done artificially by adding a dummy

"Prover 2" query to Prover 1, and vice versa. This makes the number of queries $Q = Q_1 Q_2$ for both provers. With this correction, $A_1/Q + A_2/Q \geq \log_2 m$ implies the favourable bound $A_1 + A_2 \geq Q \log_2 m$.

Summarizing, if the 3SAT formula is satisfiable then there is a set cover of size roughly $2Q$, and otherwise the minimal set cover must contain at least $Q \log_2 m$ sets. So an approximation algorithm cannot produce an approximation better than $\log_2 m/2$.

We got a gap depending on $m$, whereas we actually want a gap depending on $n = |U| = (3N)^C m$, where $N$ is the number of clauses. Since

$$\log n = C \log(3N) + \log m,$$

in order for $\log_2 m$ to be comparable with $\log_2 n$ we need that $m = (3N)^{\Omega(C)}$. Now $C$ must be big enough so that

$$\delta^C < \frac{4}{(\log_2 m)^2} = O(C \log(3N))^2.$$

Ignoring the factor of $C$ on the right, we see that $C = \Omega(\log \log N)$. Therefore $n = \Omega(N^{\log \log N})$. Playing with the epsilons, we can actually carry everything through with $n = \Theta(N^{\log \log N})$. Note that the reduction is only quasi-polynomial, and this prevents us from getting an NP-hardness result.

Let's recap our exploits: we started with a 3SAT instance of size $\Theta(N)$, and constructed a set cover instance of size $n = \Theta(N^{\log \log N})$. If the original 3SAT instance was satisfiable, then there is a set cover of size $2Q$. Otherwise, any set cover has size at least roughly $Q \log_2 n$. So if there exists a polynomial-time approximation algorithm for set cover with approximation ratio better than $\log_2 n/2$, we get an algorithm deciding 3SAT in time $\Theta(N^{\log \log N})$, and so $NP \subset \text{Time}(N^{\log \log N})$.

Taking the contrapositive, *unless NP has $N^{\log \log N}$ algorithms, there is no polynomial-time algorithm approximating set cover to within* $\log_2 n/2$. This result leaves two things to be desired: first, the correct threshold should be $\log n$ rather than $\log_2 n/2 \approx 0.72 \log n$; second, the hypothesis should be $NP \neq P$. In the next section we explain how Feige achieved the first goal. Raz and others achieved, separately, the second goal. However, so far no one has been able to achieve both simultaneously.

## 5 Perfecting the Construction

In this section we put ourselves in the shoes of Uri Feige, and show how to obtain, using a very similar construction, a bound of the form $(1 - \epsilon) \log n$ for every $\epsilon > 0$; the same technique can be used to give a bound of the form $(1 - o(1)) \log n$, which unfortunately does not match the precise performance guarantees of known algorithms.

The approximation factor of $\log_2 n/2$ derived from a construction in which the optimal set cover is of size 2, but a greedy algorithm uses $\log_2 m$ sets ($\log m \approx$

$\log n$). In the beginning of today's lecture, we showed that if the optimal set cover is of size $k$, then the greedy algorithm uses at most $k \log m$ sets; this came up as the solution $t$ of $m(1-1/k)^t = 1$, via the approximation $(1-1/k)^k \approx e$. The latter approximation gets better and better as $k$ gets larger and larger; so we are led to concoct a situation in which each partition has size $k$; for $k$ large, the "non-single-minded" way to cover a $U_{q_1,q_2}$ part of the universe will require $(1-f(k))k \log m$ sets, with $\lim_{k\to\infty} f(k) = 0$.

Having replaced 2 by $k$, we also need to enlarge the number of provers to $k$, and to give the proof system some reasonable semantics. One naive attempt would be to pair up the provers, and run several two-way proofs in parallel. The provers can cheat by being inconsistent across pairs, but it doesn't help them to increase their probability of convincing the verifier. Let's try working out this idea, and see where it breaks down.

It is easy to generalize the construction of the set system. We again have a set $S(q,a,i)$ for each possible question-answer pair for Prover $i$; we arrange so that the number of questions each prover can be asked is the same number $Q$. The universe $U$ of size $n$ is divided into pieces $U_{q_1,\ldots,q_k}$ of size $m$. Each of them can be covered by any of $2^{(k/2)C}$ partitions $U_{q_1,\ldots,q_k,\alpha,\cdot}$ of size $k$, and $S(q,a,i)$ contains a set $U_{q_1,\ldots,q_k,\alpha,i}$ whenever $q_i = q$ and $\alpha$ is consistent with $a$. If the 3SAT formula is satisfiable then we can cover $U$ with $kQ$ sets, which give the correct answer for every possible question. It remains to see what happens if the formula is not satisfiable.

Given a set cover consisting of $A_i$ sets belonging to Prover $i$, we use the same strategy as in the two-player case: for every question a prover is asked, she replies a random answer from those appearing in the set cover. On average, there will be $A_i/Q$ possible answers in her disposal. The provers convince the verifier if every pair of provers chooses two consistent answers. If this does not happen, then we have a cover of $U_{q_1,\ldots,q_k}$ using at most $k/2$ sets from each possible partition. Unfortunately, these constraints are not stringent enough, since the greedy algorithm shows that $U_{q_1,\ldots,q_k}$ can be covered using $k/2 \log_2 m$ sets, so that we get nothing new.

What we would like is a constraint enforcing the ill-behaved set cover to choose at most *one* set from each possible partition. The greedy algorithm now follows the asymptotics of the set cover greedy algorithm, namely we need $(1-f(k))k \log m$ sets to cover the $U_{q_1,\ldots,q_k}$ part of the universe (recall that $f(k) \approx 0$ for large $k$).

So we need the verifier to be convinced from any potential pair of provers. This requires, at the very least, that any two provers will have some consistency check associated with them. This is easy to achieve: for each pair $(i,j)$ of provers, have one of them tell you the values of $C$ clauses, and the other one tell you the values of $C$ selected variables, one from each clause.

The new $k$-prover system still has the property that the provers can completely convince the verifier when the 3SAT instance is satisfiable. Conversely, when the instance is not satisfiable, no pair of provers can convince the verifier with probability better than $\delta^C$. So in the former case the verifier accepts the proofs in a very strong manner, whereas in the latter case she utterly rejects them.

Wrapping up, when the 3SAT formula is satisfiable, there is a set cover of size $kQ$; otherwise, any set cover must contain at least $kQ(1 - f(k)) \log m$ sets. The rest of the proof carries through, and so we get that *unless NP is solvable in time $N^{O(\log \log N)}$, no polynomial-time algorithm can approximate set cover to within $(1 - \epsilon) \log n$, for any $\epsilon > 0$.*

Using more than $\log \log N$ parallel rounds, Feige gets an $\epsilon$ that goes down with $n$. The idea is to make $k$ grow with $N$. This requires a more efficient construction of the $k$-prover system. Feige takes a linear code of length $\ell = \Theta(C)$ and size $k$. The verifier chooses $\ell$ clauses and highlights a variable in each clause. Prover $i$ is then asked for a mix of clauses and variables, depending on her vector (say 0 means clause and 1 means variable). The fact the the code is good means that the verify can cross-check any two provers; the distance between two codewords is exactly equal to the number of consistency checks.

## 6 Bibliography

Lund, C. and Yannakakis, M. 1994. On the hardness of approximating minimization problems. *J. ACM 41, 5* (Sept.), 960–981.

Naor, M., Schulman, L. and Srinivasan, A. 1995. Splitters and near-optimal derandomization. In *FOCS 95*, 182–191.

Feige U. 1998. A Threshold of ln $n$ for Approximating Set Cover. *J. ACM 45, 4* (July), 634–652.