PCP theorem: gap amplification and hardness of independent set

CS 6810—David Steurer

Spring 2016

1 Introduction

Recall that in Max3Sat, we are given a list of clauses, each the disjuntion of three boolean literals (either a boolean variable or the negation of one), and the goal is to find an assignment to the variables that satisfies as many of the clauses as possible.

Here is an example of a Max3Sat instance φ:

$$\begin{gathered} x_1 \vee x_2 \vee \neg x_3\,, \\ \neg x_3 \vee \neg x_5 \vee x_7\,, \\ x_2 \vee x_4 \vee x_8\,, \\ \vdots \\ \neg x_7 \vee x_8 \vee \neg x_9\,, \end{gathered} \tag{1}$$

Notation: For a Max3Sat instance φ and an assignment x to the variables of φ, let $\mathrm{val}(\varphi, x)$ be the fraction of clauses in φ satisfied by x and let $\mathrm{opt}(\varphi) = \max_x \mathrm{val}(\varphi, x)$ be the maximum value of an assignment for φ.

In these notes, we will discuss the PCP theorem and its applications to hardness of approximation.

PCP theorem: There exists a polynomial-time function f that maps every Max3Sat instance φ to a Max3Sat instance φ' with the following properties:

- YES case: if $opt(\varphi) = 1$ then $opt(\varphi') = 1$
- NO case: if $opt(\varphi) < 1$ then $opt(\varphi') < 0.99$

The PCP theorem shows that distinguishing[1] between the cases $opt(\varphi') = 1$ and $opt(\varphi') < 0.99$ is as hard as deciding 3Sat. in particular, the PCP theorem shows that if $P \neq NP$ no polynomial time algorithm can achieve approximation ratio 0.99 for Max3Sat.

Aside: For Max3Sat, there is a polynomial-time algorithm that achieves an approximation ratio of 7/8. Refinements of the PCP theorem also show that no polynomial-time algorithm achieves a better approximation ratio.

Aside: The pcp theorem is not just about 3Sat and Max3Sat. Instead of 3Sat, we could choose any NP-complete problem as a starting point. Similarly, instead of Max3Sat, we could choose from a wide range of optimization problems (for example, max-independent-set, see below).

2 Strategy for proof of PCP theorem

In this course we will discuss a proof of the PCP theorem due to Dinur. The key component of that proof is the following lemma.

Lemma (amplification): There exists a polynomial-time function g that maps every Max3Sat instance φ to a Max3Sat instance φ' with the following properties:

1. YES case: If $opt(\varphi) = 1$, then $opt(\varphi') = 1$
2. NO case: If $opt(\varphi) < 1 - \varepsilon$ for some $\varepsilon > 0$, then $opt(\varphi') < \max\{0.99, 1 - 2\varepsilon\}$
3. Efficiency: $|\varphi'| \le O(1) \cdot |\varphi|$

Assuming this lemma, we can prove the PCP theorem by iterating the function g a logarithmic number of times.

2.1 Proof of PCP theorem from amplification lemma

Let t be a positive integer to be determined later. Let g be the t-fold iteration of f. Let φ be an arbitrary Max3Sat instance.

$$f\colon \underbrace{\varphi \mapsto_g \varphi_1 \mapsto_g \cdots \mapsto_g \varphi_t = f(\varphi)}_{t \text{ times}} \tag{2}$$

YES case: First suppose that $opt(\varphi) = 1$. After t iterations of the function g, we obtain a Max3Sat instance φ_t with optimal value $\mathrm{opt}(\varphi_t) = 1$.

NO case: Next suppose that $opt(\varphi) < 1$. Let $n = |\varphi|$ be the length of the binary encoding of φ. Since φ contains less then n clauses, its optimal value is less than $opt(\varphi) < (n-1)/n = 1 - 1/n$. Therefore, after t iterations of the function g, we obtain a Max3Sat instance φ_t with optimal value $opt(\varphi_t) < \max\{0.99, 1 - 2^t/n\}$.

Running time: Since g is polynomial-time computable, the running time of its t-fold iteration f is polynomial in $|\varphi| + |\varphi_1| + \cdots + |\varphi_t|$. By the efficiency property of the function g, each of these instances has size at most $O(1)^t \cdot |\varphi_0|$. Therefore, the running time of f on input x is polynomial in $t \cdot O(1)^t \cdot |x|$.

Putting it together: We choose $t = \log n$. Then, in the NO case, we have

$$opt(\varphi_t) < \max\{0.99, 1 - 2^t/n\} = \max\{0.99, 0\} = 0.99 \tag{3}$$

and the running time of g on inputs of length n is $t \cdot O(1)^t \cdot n = n^{O(1)}$. □

Aside: The strong efficiency guarantee for g is crucial. If we had a weaker efficiency guarantee, say $|g(\varphi)| \le |\varphi|^2$, we could only conclude $|\varphi_t| \le |\varphi|^{2^t}$, which is an exponential blowup for the choice $t = \log n$.

3 Hardness of approximation for independent set

Recall: We say a subset S of vertices is *independent* in a graph, if no edge of the graph has both endpoints in the set S.

The independet set problem (IndSet) is, given a graph, to find an independent set in the graph that is as large as possible.

Notation: Let $\alpha(G)$ denote maximum size of a independent set in G.

Theorem. Unless P=NP, there is no polynomial time algorithm that achieves approximation ratio 0.99 for independent set.

We show the theorem by proving the following lemma.

Lemma. There exists a polynomial time function f that maps every Max3Sat instance φ to a graph G with n vertices such that

- YES case: If $opt(G) = 1$, then $\alpha(G) = |n|/7$
- NO case: If $\text{opt}(G) < 0.99$, then $\alpha(G) < 0.99 \cdot |n|/7$

Proof sketch of lemma. We let f be the standard NP-hardness reduction from 3Sat to IndSet. Let φ be a 3cnf formula with m clauses. Let G be the graph obtained by applying the reduction f to φ.

The reduction f has the property that assignments for φ correspond to independent sets in G such that the value of the assignment is proportional to the size of the independent set.

Recall that G contains a "gadget" consisting of 7 vertices for each clause in φ (one vertex per satisfying assignment for the clauses). In particular, if φ consists of m clauses, then G has $n = 7m$ vertices. Every assignment φ corresponds to an independent set S that selects exactly one vertex out of every gadget for a satisfied clause, so that $|S| = \text{val}(\varphi, x) \cdot n/7$. Similarly, every independent set S in G corresponds to an assignment for φ that satisfies every clauses C such that S intersects the gadget of clause C. In particular, $\text{val}(\varphi, x) \cdot n/7 \geq |S|$.

It follows that $\alpha(G) = \text{opt}(\varphi) \cdot |V(G)|/7$, which implies the lemma. □

4 Gap amplification for independent set

The following theorem shows that if P $\neq$ NP then no polynomial time algorithm achieves approximation ratio 0.001 for independent set. There are refinements of this theorem that rule out approximation ratio $n^{1-\varepsilon}$ for all constants $\varepsilon > 0$.

Theorem: For every k, there exists a polynomial time computable function f such that $\alpha(f(G)) = \alpha(G)^k$ for every graph G. In particular, for $k = 1000$, the function maps every graph G on n vertices to a graph H with the following properties:

- YES case: If $\alpha(G) \geq n/7$, then $\alpha(H) \geq (n/7)^k$.
- NO case: If $\alpha(G) < 0.99 \cdot n/7$, then $\alpha(H) < 0.99^k \cdot (n/7)^k \leq 0.001 \cdot (n/7)^k$.

Proof of theorem: Let G be a graph with vertex set V. Construct H with vertex set V^k such that $(u_1, \dots, u_k) \sim_H (v_1, \dots, v_k)$ if $u_i \sim_G v_i$ for at least one $i \in [k]$. (This construction is sometimes called the k-fold OR-power of G.)

For a set $S \subseteq V^k$, let $S_i \subseteq V$ be the vertices that appear in the i-th coordinate of S.

By construction of H, the following statements are equivalent for all $S \subseteq V^k$:

- S is independent in
- $S_1, ..., S_k$ are independent in G
- $S_1 \times \cdots \times S_k \supseteq S$ is independent in H

It follows that every maximum size independent set in H is obtained as the cartesian product of maximum size independent sets in G. Therefore,

$$\alpha(H) = \alpha(G)^k . \quad \square \tag{4}$$

Footnotes

1. We say that an algorithm distinguishes two sets $A, B \subseteq \{0,1\}^*$ if the algorithm outputs YES for every instance $x \in A$ and NO for every instance $x \in B$. For instances that are neither in A nor B, it doesn't matter what the algorithm outputs. In this context, the pair (A, B) is called a *promise problem*. We say that a promise problem can be solved in polynomial time if there exists a polynomial time algorithm that distinguishes between the YES set and the NO set of the promise problem. Note that distinguishing is only possible if the YES set and NO set are disjoint.