6.893 Approximability of Optimization Problems	November 22, 1999
Lecture 20	
Lecturer: Madhu Sudan	*Scribe: John Dunagan*

Introduction

In today's lecture, we will discuss the composition of proof systems and the proof of the main PCP theorem. The theorem on the composition of proof systems will take up the majority of our dicussion even though the proof is mainly syntactic; we just check that all the symbols do go into the correct places. Composition of proof systems will be as powerful a tool for the construction of PCP's as composition of error-correcting codes was for achieving improved error-correcting codes.

Outer Verifiers

In order to combine proof systems, we are going to need to define the modules that we wish to combine very carefully. The two modules will be an **outer verifier** and an **inner verifier system**. They can be combined to produce a new outer verifier. It should be clear then that the outer verifier must actually deal with the external problem, be it MAX-CUT or MAX-SAT or TSP. The inner verifier system will deal only with a single canonical problem, that of algebraic circuit satisfiability, which we denote by CKT-SAT.

Both the outer and inner verifiers are defined in terms of a bunch of parameters restricting how they can interact with a prover. Hereafter we will just refer to a prover as an oracle since a prover in one setting will be part of a verifier in the next.

Definition 1 *V is an $(r(n), p, a(n))$-outer verifier if V takes an instance x of length $\leq n$ and*

1. *computes $\leq p$ probes and a circuit C of size $O(p \cdot a(n))$ using $\leq r(n)$ random bits*
2. *asks the probes non-adaptively to an oracle Π*
3. *only accepts responses from Π that are strings of length $\leq a(n)$*
4. *decides whether to accept or not based on whether the responses from Π satisfies C*

Let's discuss this definition. We neglected a few obvious requirements, like V being a polynomial time probabilistic turing machine. This definition differs from that in [ALMSS92] in several ways. For one, we do not specifying the circuit size used to compute the final decision as an explicit parameter - instead, it is required to be linear in the size of the response from the oracle. Also, we do not allow the number of probes to be a function of n.

In terms of terminology, a probe is a question asked of the oracle that has some specified length, while a query is a question that is answered by a single bit. It is common to say that the query size is equal to the number of probes times the answer size.

Next, V is required to figure out all the probes it will ask and what it will do with the answers before asking any of the probes to an oracle. Simple enough. However, in order to keep track of the parameters that soon start flying around, we will adopt the following representation of V.

V

randomness	$= r(n)$
probe no.	$= p$
answer size	$= a(n)$

An (r, p, a)-outer verifier is often referred to as an (r, p, a)-restricted verifier since we are restricting how many questions the verifier can ask, how much randomness it can use, etc. From our previous classes, the relation between the existence of appropriately restricted verifiers and language membership should be clear.

$$L \in rPCP_{c,s}[r,p,a] \Leftrightarrow \exists\ (r,p,a)\text{–restricted verifier } V \text{ such that}$$
$$x \in L \Rightarrow \exists \Pi \text{ s.t. } \Pr\left[V^{\Pi}(x,R)\right] \geq c$$
$$x \notin L \Rightarrow \forall \Pi \ \Pr\left[V^{\Pi}(x,R)\right] \leq s$$

Here rPCP stands for restricted PCP. The above statement is just a declaration of notation. Parsing the statement into english, we would say "L is a language in the class of rPCP languages with completeness c, soundness s and such that the restricted verifier uses no more than $r(n)$ random bits to make $p(n)$ queries, each with answer size $a(n)$." The probabilities in the statements are taken over the choice of random string R or length $r_1(n)$, and V is understood to be a deterministic turing machine once R is specified. The notation V^{Π} is simply to denote that V treats Π as the oracle to whom it asks questions.

The important conceptual changes from the last class (when we discussed MIP) are that the questions are non-adaptive, we are counting the amount of randomness that the verifier requires, and the verdict is computed by a linear sized circuit (linear in the total size of the responses $p \cdot a$).

Inner Verifiers

An inner verifier system consists of an inner verifier and an encoding/decoding scheme. When we speak of a (k, r, p, a)–inner verifier system, we mean a (k, r, p, a)–inner verifier along with an encoding scheme E and a decoding scheme D.

An inner verifier takes an instance of CKT-SAT consisting of a circuit of size n with k multi-bit inputs. (This is the origin of the new parameter, k.) The verifier will interact with $k + 1$ separate oracles, not just one. Formally, we have

Definition 2 *V is a $(k, r(n), p, a(n))$-inner verifier if V takes a circuit C of length $\leq n$ and*

1. *computes $\leq p$ probes and a circuit C' of size $O(p \cdot a(n))$ using $\leq r(n)$ random bits*
2. *asks the questions non-adaptively to a series of oracles $\Pi_1, ..., \Pi_{k+1}$*
3. *only accepts responses from the oracles that are strings of length $\leq a(n)$*
4. *decides whether to accept or not based on whether the responses satisfy C'*

Conceptually here, $k \leq p$. We never bother having more oracles around than probes. We also pause to note that the MIP verification system is this with $k = 1$. As before, we have a notational picture for V

V

oracle no.	$= k$
randomness	$= r(n)$
probe no.	$= p$

answer size $= a(n)$

We now specify the completeness and soundness conditions for the inner verifier; this is where the encoding and decoding schemes (E and D) make their entrance. For the inner verifier system to have completeness c, we require that

Definition 3 *Completeness: If a circuit C is satisfied by $x_1, ...x_k$, and $\Pi_i = E(x_i)$, then $\exists \Pi_{k+1}$ such that*

$V^{\Pi_1,...\Pi_{k+1}}(C, r)$ *accepts with probability* $\geq c$

where r specifies the random string and the probability is taken over r.

Note that in this definition Π_{k+1} may be a function of x_1 through x_k, but Π_i is a function of x_i alone $(1 \leq i \leq k)$. In order for the inner verifier system to have soundness s, we require that

Definition 4 *Soundness: If $\exists \Pi_1, ...\Pi_{k+1}$ such that*

$V^{\Pi_1,...\Pi_{k+1}}(C, r)$ *accepts with probability* $\geq s$

then the circuit C is satisfied by $x_1, ...x_k$, where $x_i = D(\Pi_i)$

We note that encodings and decodings here are allowed to be inefficient - they may take exponential time, since they are computer by the prover.

Composition Theorem [Arora and Safra]

Theorem 1 *If language $L \in rPCP_{1,1-\epsilon}[r_1(n), p_1, a_1(n)]$ and $\exists$ a $(k, r_2(n), p_2, a_2(n))$-inner verifier system with completeness 1 and soundness $1 - \epsilon$, where $k = p_1$, then*

$L \in rPCP_{1,1-\epsilon^2}[r_1(n) + r_2(a_1(n)), p_2, a_2(a_1(n))]$

We will prove this in a later section. Let us first note the consequences of this theorem when we apply it to inner and outer verifiers that we have already constructed.

Consequences

Any NP-complete language L has an outer verifier V with the following parameters

V

randomness	$= 0$
probe no.	$= 1$
answer size	$= n$

The first inner verifier we recall from lecture 17 (although we deferred part of the proof of its existence to lecture 18) and we label it $V1$

$V1$

$$\begin{array}{rl} \text{oracle no.} & = k \\ \text{randomness} & = O(\log n) \\ \text{probe no.} & = k+3 \\ \text{answer size} & = (\log n)^{O(1)} \end{array}$$

combining V and $V1$ yields an outer verifier V' with parameters

V'

$$\begin{array}{rl} \text{randomness} & = O(\log n) \\ \text{probe no.} & = 4 \\ \text{answer size} & = (\log n)^{O(1)} \end{array}$$

combining V' and $V1$ yields an outer verifier V'' with parameters

V''

$$\begin{array}{rl} \text{randomness} & = O(\log n) + O(\log(\log n)^{O(1)}) = O(\log n) \\ \text{probe no.} & = 7 \\ \text{answer size} & = (\log(\log n)^{O(1)})^{O(1)} = (\log\log n)^{O(1)} \end{array}$$

As prophesied, we have obtained an outer verifier with better parameters than either one of our proof systems had on its own. Another inner verifier system that we saw in lecture 17 we denote by $V2$. It had parameters

$V2$

$$\begin{array}{rl} \text{oracle no.} & = k \\ \text{randomness} & = O(n^3) \\ \text{probe no.} & = k+23 \\ \text{answer size} & = 1 \end{array}$$

Combining V'' with $V2$ yields an outer verifier V''' with parameters

V'''

$$\begin{array}{rl} \text{randomness} & = O(\log n) + poly(\log\log n) = O(\log n) \\ \text{probe no.} & = 30 \\ \text{answer size} & = 1 \end{array}$$

As we have noted in previous classes, the construction of this PCP shows the inapproximability of MAX-SAT. We ignored the soundness error that was building up at each step, and putting this back in yields our final result

$L \in rPCP_{1,1-\epsilon^8}[O(\log n), 30, 1]$

Proof of Composition Theorem

Let V_1 be an outer verifier and V_2 be an inner verifier. We will construct V, our composed verifier. Let x be the instance of our problem and Π the proof of language membership that we are supposed to probabilistically query.

On input x, V first emulates V_1. V_1 tosses $r_1(n)$ coins and decides what $p_1(n)$ probes to ask oracle Π. It then builds circuit C of size $p_1 \cdot a_1(n) = O(a_1(n))$ that specifies the acceptance procedure.

V then emulates V_2. V_2 now just wants to determine the satisfiability of C. V_2 tosses its random string R_2 of length $r_2(O(a_1(n)))$ and decides which p_2 probes to ask oracles $\Pi_1 ... \Pi_{k+1}$

V asks the probes specified by V_2 to Π and accepts if V_2 does.

Here we include what will hopefully be an informative picture to aid in the understanding of the oracle Π. We first draw the oracle Π' that V_1 queries, and then the construction of Π from Π'

$$
\begin{array}{ccc}
 & \text{the oracle } \Pi' & \text{the oracle } \Pi \\
 & x_1 & E(x_1) = \Pi_1 \\
V_1 \longrightarrow & x_2 & E(x_2) = \Pi_2 \\
 & x_3 & E(x_3) = \Pi_3 \\
 & \cdots & \cdots \\
 & & \Pi_{k+1}^{R}
\end{array}
$$

As demonstrated in this picture, the final series of sub-oracles in Π does not depend on any of the entries in Π'. Instead, it is a function of the random string R, and hence there is one for every possible random string R.

Now that we have specified the behavior of V, we have to prove that it fulfills the completeness and soundess requirements.

Completeness: suppose $x \in L$. The completeness condition for V_1 implies that there is an oracle Π' causing V_1 to accept with probability 1. We will convert Π' into an oracle Π that causes V to accept with probability 1. We think of Π' as a large table, a list of all the responses it gives to different questions. Construct Π by replacing each entry e of Π' by $E(e)$ (the encoding of e). From V_2's point of view, $E(e)$ is an oracle, not just a string, and the oracle has answer size $a_1(n)$. Thus Π is an oracle with lots of sub-oracles.

Let us do a reality check to make sure that this does indeed make sense. The completeness condition on V_2 was that if the inputs to the circuit C caused C to accept, then the encodings of the inputs are oracles causing V_2 to accept. So we are doing the straightforward thing.

This covers the oracles Π_1 through Π_k that V_2 queries. Let us lastly consider Π_{k+1}. For each possible random string $R_1 \in \{0,1\}^{r(n)}$, let C be the circuit generated by V_1. Let $a_1, ... a_p$ be the responses given by Π' when queried by V_1. Then the completeness condition on V_1 implies that $a_1, ... a_p$ is a satisfying assignment for C. Then the completeness condition on V_2 implies that there exists an oracle τ such that

$$\Pr\left[V_2^{E(a_1),\ldots E(a_k),\tau}(C, R_2) = \text{accept}\right] = 1$$

where the probability is taken over all random strings R_2. Let this sequence of τ's (one for every random string R_1) be the remaining sub-oracles of Π. Then V_2 accepts with probability 1. Thus Π causes V to accept with probability 1 also.

Soundness: suppose $x \notin L$. We will assume there is an oracle Π causing V to accept with probability greater than ϵ^2 and derive that there is an oracle Π' causing $V1$ to accept with probabliity greater than ϵ, contradicting the soundness condition for V_1.

First consider the set of random strings $\mathcal{R}_1$ defined by

$$\mathcal{R}_1 = \{R_1 \in \{0,1\}^{r_1(n)} \mid \Pr\left[V^{\Pi}(x,(R_1,R_2))= \text{accept}\right] \geq \epsilon\}$$

where the probability is taken over random strings $R_2 \in \{0,1\}^{r_2(a_1(n))}$. We first note that $\mathcal{R}_1$ constitutes at least an ϵ fraction of all the strings in $\{0,1\}^{r_1(n)}$. To see this, consider the acceptance probability if it were not the case. We find that the acceptance probability of V is
$\leq \Pr[R_1 \in \mathcal{R}_1] \cdot 1 + \Pr[R_1 \notin \mathcal{R}_1] \cdot \epsilon = (\epsilon)1 + (1-\epsilon)\epsilon = 2\epsilon - \epsilon^2 \geq \epsilon^2$ where the last inequality follows from the fact that ϵ is a number between 0 and 1.

We now restrict our attention to showing that $\forall R_1 \in \mathcal{R}_1$, V_1 accepts. This will be enough to contradict the soundness of V_1.

Consider all the suboracles in Π that do not correspond to a τ from our discussion of completeness. Denoting the i^{th} sub-oracle of Π' by O_i, let Π consist of the strings $D(O_i)$. Since we are now restricting our attention to R_1 such that V_2 rejects with probability $\leq \epsilon$, then by the soundness condition for V_2 we have that every one of the decoded strings corresponds to part of a satisfying assignment for the circuit we are considering. Since the appropriate locations in Π' contain exactly these decodings, we find that $V_1^{\Pi'}(x, R_1) = \text{accept}$, contradicting the soundness of V_1.

The last thing to note is that our new outer verifier does indeed have the claimed parameters with respect to randomness, etc. This completes the proof.

Construction of Inner Verifiers

To be filled in by Madhu.