Probabilistically Checkable Proofs and Hardness of Approximation

León Illanes F.

June 5, 2012

Introduction

Motivation for PCP

Suppose somebody wants to convince you that a given propositional formula is satisfiable. Usually this works in the following way:

- He sends you a certificate (a satisfying assignment)
- You substitute it into the formula

This requires that you check the whole certificate.

Motivation for **PCP** (cont.)

Probabilistically Checkable Proofs provide the following alternative:

- He sends you a modified version of the certificate
- You randomly (probabilistically) select a few locations of the certificate
- You check with those parts in such a way that if the formula is unsatisfiable you will reject with high probability

"few locations" = constant number of bits

Motivation for Approximation

One of the main motivations of studying **NP**-completeness has been to understand the complexity of many optimization problems (eg. TSP).
If $\mathbf{P} \neq \mathbf{NP}$, **NP**-hard optimization problems cannot be solved efficiently.

What happens if we care merely for an approximate solution?

Motivation for Approximation (cont.)

Interesting questions:

- What are the best possible efficient approximation algorithms for **NP**-hard optimization problems?
- Can we approximate these problems to within arbitrary precision and still be efficient?

This could mean that **P** vs. **NP** would have little practical importance

Approximation Algorithms

Let's begin with an example: having 3SAT as the basic example of **NP**-complete decision problems, we will produce a corresponding optimization problem.

Definition (MAX-3SAT)

Given a 3CNF propositional formula φ, find an assignment μ that maximizes the number of satisfied clauses.

MAX-3SAT

Definition (MAX-3SAT)

Given a 3CNF propositional formula φ, find an assignment μ that maximizes the number of satisfied clauses.

It is immediate that MAX-3SAT is **NP**-hard[1]. Given an assignment μ that solves MAX-3SAT, to decide 3SAT we have only to verify whether μ satisfies φ.

[1] Formally MAX-3SAT $\in$ **NPO**, **NP**-hard is for decision problems.

Approximating MAX-3SAT

The following algorithm finds an assignment that satisfies at least half the clauses of a formula:

Input: the set of clauses Φ

for all variable v **do**
    Assign to v the value that results in satisfying the greater number of clauses in Φ.
    Remove from Φ all the clauses that have already been satisfied.
    if $\Phi = \varnothing$ **then**
        Assign anything to the remaining variables
        return
    end if
end for

Approximating MAX-3SAT (cont.)

Exercise

Let $\varphi = (\bar{v_1} \vee v_2 \vee v_3) \wedge (\bar{v_2} \vee v_3 \vee v_4) \wedge (v_1 \vee \bar{v_3} \vee v_4) \wedge (v_1 \vee \bar{v_2} \vee \bar{v_4})$
$\wedge (v_2 \vee \bar{v_3} \vee \bar{v_4}) \wedge (\bar{v_1} \vee v_3 \vee \bar{v_4}) \wedge (v_1 \vee v_2 \vee v_3) \wedge (\bar{v_1} \vee \bar{v_2} \vee \bar{v_3}) \wedge (v_1 \vee v_2 \vee \bar{v_4})$.
Run the algorithm!

Approximation for MAX-3SAT: Definitions

Given a propositional formula φ, we define its **value**:

Definition (value of a formula)

The **value** of a formula φ, denoted $val(\varphi)$, corresponds to the maximum fraction of clauses of φ that can be satisfied by one assignment to its variables. In particular, φ is satisfiable iff $val(\varphi) = 1$.

With this, we can define an approximate solution to φ:

Definition (ρ-approximate solution to a formula)

For every $\rho \leq 1$, an assignment μ is said to be a ρ-approximate solution for a formula φ with m clauses if it satisfies at least $\rho \cdot val(\varphi) \cdot m$ of φ's clauses.

Approximation for MAX-3SAT: Definitions (cont.)

Given these definitions, we can proceed to define what an approximation algorithm for MAX-3SAT should be:

Definition (ρ-approximation algorithm for MAX-3SAT)

For every $\rho \leq 1$, an algorithm A is a ρ-approximation algorithm for MAX-3SAT if for every 3CNF formula φ, $A(\varphi)$ returns a ρ-approximate solution to φ.

Our algorithm corresponds to a $\frac{1}{2}$-approximation algorithm for MAX-3SAT. **Can we improve it?**

There's a hard limit: $\frac{7}{8}$-approximation

Probabilistically Checkable Proofs

The idea is to have a proof system for decision problems in which you receive a membership certificate that can be verified by probabilistically selecting a constant number of locations. The system must satisfy the following criteria:

- A correct certificate will never fail to convince you
- You will reject falsely claimed certificates with high probability

A PCP example

- Let A be any axiomatic system for mathematics in which proofs can be verified deterministically in polynomial time (regarding the length of the proof).
- $L = \{\langle \varphi, 1^n \rangle | \varphi$ has a proof in A of length $\leq n\}$ is in **NP**.
- A **PCP** system would allow probabilistically checkable "proofs" for any provable mathematical statement. These proofs could be checked by examining only a constant number of bits.
- "New shortcut found for long math proofs!"

A definition for **NP**

A language L is in **NP** if there is a polynomial time Turing Machine V (the "verifier") that, given input x, verifies a certificate that proves that $x \in L$.

Definition ($L \in \mathbf{NP}$)

- $x \in L \Rightarrow \exists \Pi \;\; V^{\Pi}(x) = 1$
- $x \notin L \Rightarrow \forall \Pi \;\; V^{\Pi}(x) = 0$

Where V^{Π} denotes the verifier with access to certificate Π.

A definition for **PCP**

PCP generalizes the previous notion.

- It uses a probabilistic verifier
- The verifier has *random access* to the certificate Π
  Random access as in RAM:
  - The TM has a special *address tape* where it can write a number i and then somehow receive the bit $\Pi[i]$

Interesting considerations

- Address size is logarithmic in the proof size: a polynomial time verifier can check a certificate of exponential length
- We can define two different types of verifiers:
  - *Nonadaptive* verifiers select which bits of the certificate to read based only on the input and random tape
  - *Adaptive* verifiers may select bits based on the results of previous readings

We will restrict **PCP** verifiers to be nonadaptive.

Formal definition

Definition ((r, q) – PCP verifier)

Let L be a language and $q, r : \mathbb{N} \to \mathbb{N}$. We say that L has an $(r(n), q(n))$ – **PCP** verifier if there exists a polynomial time probabilistic Turing Machine V that is:

- **Efficient:** On input $x \in \{0,1\}^n$ and given random access to $\Pi \in \{0,1\}^{q(n)2^{r(n)}}$, V uses at most $r(n)$ random coins and makes at most $q(n)$ queries to Π. It then outputs 1 or 0 for "accept" or "reject", respectively.
- **Complete:** If $x \in L$, then there exists Π such that $\Pr\left(V^{\Pi}(x) = 1\right) = 1$.
- **Sound:** If $x \notin L$, then for every Π we have $\Pr\left(V^{\Pi}(x) = 1\right) \leq \frac{1}{2}$.

Complexity class

Definition (PCP $(r(n), q(n))$)

We say that a language L is in **PCP** $(r(n), q(n))$ if there are some constants $c, d > 0$ such that L has a $(c \cdot r(n), d \cdot q(n))$ – **PCP** verifier.

The **PCP** Theorem

The **PCP** Theorem says that for every languange in **NP**, there exists a highly efficient **PCP** verifier.

Theorem (PCP)

$\mathbf{NP} = \mathbf{PCP}\,(\log(n), 1)$

Proof? Long, long, long...

Interesting considerations

- For $q(n) \cdot 2^{r(n)}$ random bits, we need $r(n) + \log q(n)$ address bits.
- $\mathbf{PCP}\,(r(n), q(n)) \subseteq \mathbf{NTIME}(2^{O(r(n))}q(n))$
  A nondeterministic machine could guess the proof in time $2^{O(r(n))}q(n)$ and then simulate for all $2^{O(r(n))}q(n)$ possible random coin tosses.
- A special case of the above is $\mathbf{PCP}(\log(n), 1) \subseteq \mathbf{NTIME}(2^{O(\log(n))}) = \mathbf{NP}$. This proves one direction of the theorem.
- For $x \notin L$, the verifier rejects every proof with probability at least $\frac{1}{2}$. This is difficult to prove.
- The constant $\frac{1}{2}$ for the soundness requirement is arbitrary.

Example: GNI

Graph nonisomorphism is in **PCP** $(poly(n), 1)$.
Let the input be (G_0, G_1), where both G_0 and G_1 have n nodes.

Π contains for each possible graph H of n nodes, a bit that represents whether $G_0 \equiv H$ or $G_1 \equiv H$ (if neither is true, the value can be random).

The verifier picks $b \in \{0, 1\}$ at random and a random permutation. It applies the permutation to the vertices of G_b to obtain an isomorphic graph, H. It then queries the corresponding bit of Π and accepts iff the bit is b.

Hardness of Approximation

Theorem

There exists $\rho < 1$ *such that for every* $L \in \boldsymbol{NP}$ *there is a polynomial time function* f *mapping strings to 3CNF formulae so:*

- $x \in L \Rightarrow val(f(x)) = 1$
- $x \notin L \Rightarrow val(f(x)) < \rho$

Hardness of Approximation (cont.)

From the theorem, we can devise a way to transform a ρ-approximation algorithm for MAX-3SAT into an algorithm that decides L.

We have that $x \in L \Leftrightarrow A(f(x)) > \rho m$, where m is the number of clauses in $f(x)$.

We obtain the following corollary.

Corollary

There exists $\rho < 1$ such that if there is a polynomial time ρ-approximation algorithm for MAX-3SAT, then $\boldsymbol{P} = \boldsymbol{NP}$.

Relationship between PCP and HoA

We will now show the relationship between the PCP Theorem and the theorem about Hardness of Approximation.
We will use the notion of *Constraint Satisfaction Problems*.
This is a generalization of 3SAT that basically allows clauses of arbitrary form (not just CNF).

CSP

Definition (Constraint Satisfaction Problems)

- For $q \in \mathbb{N}$, a q^{CSP} instance φ is a collection of functions $\varphi_1, \cdots, \varphi_m$ (called constraints) from $\{0,1\}^n$ to $\{0,1\}$ such that each function φ_i depends on at most q of its input locations.
- We say that an assignment $\mu \in \{0,1\}^n$ satisfies constraint φ_i is $\varphi_i(\mu) = 1$.
- The fraction of constraints satisfied by μ is $\frac{\sum_{i=1}^{m} \varphi_i(\mu)}{m}$ and we let $val(\varphi)$ denote the maximum of this value over all μ's.
- We say that φ is satisfiable if $val(\varphi) = 1$.
- We call q the arity of φ.

CSP (cont.)

Note that CSP generalizes 3SAT. Consider 3^{CSP} in which every φ_i is an OR of the relevant literals.

Gap CSP

The following theorem can be proven to be equivalent to the PCP Theorem:

Theorem (ρ-Gap q^{CSP})

*There exists constants $q \in \mathbb{N}$, $\rho \in (0,1)$, so that for every language $L \in$ **NP** there is a polynomial time function f mapping strings to q^{CSP} instances that satisfy:*

Completeness: $x \in L \Rightarrow val(f(x)) = 1$

Soundness: $x \notin L \Rightarrow val(f(x)) < \rho$

The following slides provide a proof of the statement:

Theorem

*The **PCP** Theorem is equivalent to the ρ-Gap q^{CSP} Theorem.*

ρ-Gap $q^{\text{CSP}} \equiv \mathbf{PCP}(\log n, 1)$

1. **PCP** $\Rightarrow$ ρ-Gap q^{CSP}
   - Let us assume the **PCP** Theorem. Specifically: $\mathbf{NP} \subseteq \mathbf{PCP}(\log n, 1)$.
   - 3SAT has a **PCP** system in which the verifier V makes a constant number q of queries and uses $c \log n$ random coins, for some constant c.
   - Given every input x and random string $r \in \{0,1\}^{c \log n}$, let $V_{x,r}$ be the function that for input Π outputs 1 if V will accept the proof Π on input x and coins r. Note that $V_{x,r}$ depends on at most q locations.

ρ-Gap $q^{\text{CSP}} \equiv \mathbf{PCP}(\log n, 1)$

- For every x, $\varphi = \{V_{x,r}\}_{r \in \{0,1\}^{c \log n}}$ is a polynomial sized q^{CSP} instance.
- Given V runs in polynomial time, the transformation of x to φ can also be carried out in polynomial time.
- Finally, if $x \in 3\text{SAT}$, then $val(\varphi) = 1$. If $x \notin 3\text{SAT}$, then $val(\varphi) \leq \frac{1}{2}$.

ρ-Gap $q^{\text{CSP}} \equiv \mathbf{PCP}(\log n, 1)$

2. ρ-Gap $q^{\text{CSP}} \Rightarrow$ **PCP**
   - Suppose that ρ-Gap q^{CSP} is satisfied for some constants q, ρ.
   - This easily translates to a **PCP** system with q queries, ρ soundness and logarithmic randomness for any language L.
   - Given input x, the verifier will run the reduction $f(x)$ to obtain a q^{CSP} instance $\varphi = \{\varphi_i\}_{i=1}^m$.

ρ-Gap $q^{\text{CSP}} \equiv \mathbf{PCP}(\log n, 1)$

- The verifier expects Π to be an assignment to the variables of φ, which it will verify by choosing a random $i \leq m$ and checking that φ_i is satisfied (q queries).
- If $x \in L$, the verifier will accept with probability 1. If $x \notin L$, then the verifir will accept with probability at most ρ.

Hardness of Approximation

The similiarities between Hardness of Approximation and ρ-Gap q^{CSP} are easy to see. Again, it is possible to prove they are equivalent. This proves that the Hardness of Approximation theorem and the **PCP** Theorem are in fact equivalent.

Bibliography

- S. Arora and B. Barak.
  Computational Complexity: A Modern Approach.
  Cambridge Univeristy Press, 2009.
- S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy.
  Proof verification and hardness of approximation problems.
  Proc. 33rd Annual Symposium on Foundations of Computer Science, 1992.
- S. Arora and S. Safra.
  Probabilistic checking of proofs: A new characterization of NP.
  Journal of the ACM, 1998.

Probabilistically Checkable Proofs and Hardness of Approximation

León Illanes F.

June 5, 2012