PCP Theorem and Hardness of Approximation
An Introduction

Lee Carraher and Ryan McGovern

Department of Computer Science
University of Cincinnati

October 27, 2003
Assuming $NP \neq P$, there are many problems that would take a prohibitively long amount of time to solve for a large number of inputs. Approximate solutions are adequate for many real world problems. Unfortunately, there are not many approximate solutions that are any better than the complexity of the original NP problem for most problems. Furthermore, Cook-Levin-Karp reductions of NP problems to NP problems with good approximate solutions, do not yield equivalent results for the reduced problem.
Outline

- Approximate Solutions
- Two Views of PCP Theorem
- Equivalence of the two Views
- Hardness of Approximation for Vertex Cover and Independent Set
- Hadamard-Walsh Transformation
- Quadratic Equation Proof using the Hadamard-Walsh Transform
PCP - Probabilistically Checkable Proofs

- **View. 1:** constructs a locally testable proof system, that is checkable by only looking at a few of the symbols of the proof.
- **View. 2:** regarding the hardness of approximating NP-Complete optimization problems.
As mentioned in the introduction, if we can find approximate solutions to NP problems that are within an acceptable bound of the exact solution, then the complexity issue of NP problems would not be so bad.

Also, finding the limits of these approximate algorithms, would be an interesting problem.

The attempt to discover the limits in turn motivated PCP Theorem

Consider two NP-Hard Optimization Problems

- Max-3Sat
- Vertex Cover
Definition 11.1

(Approximation of MAX-3SAT) For every 3CNF formula \( \varphi \), the value of \( \varphi \), denoted by \( \text{val}(\varphi) \), is the maximum fraction of clauses that can be satisfied by any assignment to \( \varphi \)'s variables. In particular, \( \varphi \) is satisfiable iff \( \text{val}(\varphi) = 1 \)

For every \( \rho \leq 1 \), an algorithm \( A \) is a \( \rho \)-approximation algorithm for MAX-3SAT if for every 3CNF formula \( \varphi \) with \( m \) clauses, \( A(\varphi) \) outputs an assignment satisfying at least \( \rho \times \text{val}(\varphi)m \) of \( \varphi \)'s clauses.
The 1/2 Approximation algorithm for Max-3Sat is relatively trivial.

**Approximate Max-3Sat Algorithm**
- for each variable $v$:
  - assign $v$ s.t. it satisfies at least 1/2 of the clauses that it appears in
  - remove all clauses satisfied by $v$

Naturally this results in satisfying at least half of the clauses, and therefore it is at least a 1/2 approximation of the actual solution.

Another approximate algorithm utilizes techniques of semidefinite programming, in which the solution is cast as the objective function of a continuous convex optimization. The solution is then chosen as the nearest integer solution to the real convex solution.
Min Vertex Cover

L. Carraher and R. McGovern
PCP Theorem and Hardness of Approximation
Definition

A minimum vertex cover of a graph G is a set c of vertices from C, such that each edge of G is incident to at least one vertex in c. The minimum solution set $m = \arg\min |c| \in C$.

The $1/2$ approximate solution relaxes the minimum requirement such that the cardinality of $m'$ has to be at most $2x$ the optimal solution set’s cardinality.
Approximate Min-Vertex-Cover

- $S = \{\}$
- while $E \neq \{\}$
  - pick any $\{u, v\} \in E$
  - $S = S \cup \{u, v\}$
  - remove edges incident to either $u$ or $v$
Approximate Min-Vertex-Cover Algorithm

- $S = {}$
- Choose edge between 6 and 4

- $S = \{6, 4\}$
- Remove edges incident to 6 and 4

L. Carraher and R. McGovern

PCP Theorem and Hardness of Approximation
Approximate Min-Vertex-Cover Algorithm

- $S = \{6, 4\}$
- choose edge between 3 and 2

remove edges incident to 3 and 2

$S = \{6, 4, 3, 2\}$
Approximate Min-Vertex-Cover Algorithm

- $S = \{6, 4, 3, 2\}$
- choose edge between 5 and 1

- remove edges incident to 5 and 1
- $S = \{6, 4, 3, 2, 5, 1\}$
Approximate Min-Vertex-Cover Algorithm

- $S = \{6, 4, 3, 2\}$
- choose edge between 5 and 1

We are done!

- $S = \{6, 4, 3, 2, 5, 1\}$
- note the cardinality is 6, the optimal solution $S = \{4, 1, 2\}$, is 3
- which is a $2\times$ approximation
Can we do better?

naturally the next question is, can we do better?
some optimizations:

- choose nodes with the greatest cardinality first
- use Depth first search, the cover is all nodes above the lowest level

This search for optimizations for both algorithms, and bounding of the approximate solution of such algorithms leads naturally into the first definition of PCP theorem.
Two Definitions of the PCP Theorem

All Languages \( L \subseteq NP \) have a PCP system wherein on input \( x \in 0, 1^n \):

- Prover \( P \) writes down a \( \text{poly}(n) \)-bit-length proof.
- Verifier \( V \) looks at and does polynomial time deterministic computation.
- Then \( V \) uses \( O(\log n) \) bits of randomness to choose \( C \) random locations in the proof. Here \( C \) is a constant.
- \( V \) reads the \( C \) randomly chosen locations from the proof and performs a test on them accepting or rejecting.
- \( V \) accepts if \( x \in L \) with probability 1. This is Completeness.
- \( V \) rejects for all \( x \in L \) with probability at most \( 1/2 \). This is Soundness.

**Theorem (11.5)**

\[ NP = PCP(\log n, C) \]
Two Definitions of the PCP Theorem

- Some false statements will be accepted
- No true statements will be rejected
- if you use $O(\text{polyn})$ bits instead of log you have $L \subseteq NEXP$
- The $1/2$ is arbitrary
The number of random bits used can be a little as three with imperfect completeness.

Specifically, for every constant $\sigma > 0$ there is a poly-size PCP for NP that reads 3 random bits and tests their XOR; it has completeness $1 - \sigma$ and soundness $1/2 + \sigma$. 

L. Carraher and R. McGovern

PCP Theorem and Hardness of Approximation
Two Definitions of the PCP Theorem

- If one NP-complete problem can produce a Polynomial Checkable Proof then by classical theory of we can create a PCP system for any NP-Complete problem.
- This is done by doing the reduction into the first step of the verifier and the prover.
- Key note that PCP does not inherently reduce the time to verify a proof due to the time used in the first step.
Of the two characterization of PCP theorems the Hardness of approximation results are considered more useful.

The Hardness of approximation results usually use problems in terms of "what is the best" or "minimum" of normal NP problems.
A key problem is MAX-3SAT which outputs the maximum fraction of clauses in a 3CNF Formula that can be satisfied.

- MAX-3SAT has no Polynomial Time Approximation Scheme for any $\sigma > 0$ where $\sigma$ is the approximation factor iff $P \neq NP$

This discovery is important because it means that approximation for some problems is not any better than solving the problem. However for other problems it is easier as there is a good approximation algorithm for the KNAPSACK problem.
The key to the proofs of the two views is a problem called CSP (Constraint Satisfaction Problems)

- CSP generalizes 3SAT by allowing clauses of arbitrary form
- CSP allows clauses that depend on more than three variables
- A $q$CSP instance is made up of $m$ functions $\phi_1, \ldots, \phi_m$
- Each function takes an input $\{0,1\}^n \rightarrow \{0,1\}$ and depends on at most $q$ variables
- 3SAT is the sub case of $q$CSP where $q = 3$ and the constraints are OR’s of the involved literals
We now define Gap (CSP) such that for every $q \in \mathbb{N}$, $p \leq 1$ define $p - \text{GAP}_q \text{CSP}$ to be the problem of determine for a given $q$CSP-instance $\phi$ weather.

- $val(\phi) = 1$ in which case we say $\phi$ is a YES instance of $p - \text{GAP}_q \text{CSP}$ or
- $val(\phi) < p$ in which case we say $\phi$ is a NO instance
- Completeness: $x \in L \Rightarrow val(f(x)) = 1$
- Soundness: $x \notin L \Rightarrow val(f(x)) < p$

This satisfies the completeness and soundness requirements for PCP.

**Theorem (11.14)**

There exist constants $q \in \mathbb{N}$, $0 < p < 1$ such that $p - \text{GAP}_q \text{CSP}$ is NP-HARD
Assume that \( NP \subseteq PCP \) will will show that \( 1/2 - GAPqCSP \) is NP-hard for some constant \( q \).

All that is needed is to reduce a NP-complete problem to \( 1/2 - GAPqCSP \) for some constant \( q \).

Given our assumption 3SAT has a PCP system that will make a constant number of queries \( C \). We use \( q = C \) and \( r \in \{0, 1\}^{c \log(n)} \).

Let \( V_{x,r} \) be the function that on a input proof outputs 1 if the verifier will accept the proof on input \( x \) and coins random bits \( r \).

Note that \( V_{x,r} \) depends on at most \( q \) locations.

Let \( \phi = \{ V_{x,r} \}_{r \in \{0,1\}^{c \log(n)}} \) and see how this is a polynomial size \( qCSP \) instance.

This \( qCSP \) instance will work for every \( x \in \{0, 1\}^n \).

Since \( V \) is a valid PCP system it will satisfy for all valid proofs and reject invalid proofs with \( p \leq 1/2 \).
If you assume that $p - GAPqCSP$ is NP hard for some constants $q \geq 1$, $p < 1$ we need to only translate this into a PCP system with $q$ queries and $p$ soundness.

This can be done by simply using the assignment of variables of $\phi$ as the proof and checking a random statement.

It will then verify this by making $q$ queries to the proof (which is all that is needed of a $q$CSP).

Obviously if the proof is valid we will not say it is invalid ever.

However if the proof is not valid we will accept it with probability at most $p$. 
Since 3CNF formulas are a special case of 3CSP as stated above 11.9 implies 11.14
We must proof 11.14 implies 11.9
This is done by mapping a \((1 - \epsilon) - \text{GAP}q\text{CSP} \phi\) to a 3SAT formula using an intermediate \(q\text{CSP} \phi'\)
We let each constraints of \(\phi\) be in CNF form of at most \(2^q\) clauses
\(\phi'\) is a collection of constraints where each constraint is a single clause in \(\phi\)
since each constraint in \(\phi\) is of at most \(2^q\) clauses then \(\phi'\) will have \(m * 2^q\) constraints where \(m\) is the number of constraints in \(\phi\)
we now reduce $\phi'$ to using the cook level reduction on each clause to 3SAT and create a new qCSP $\phi''$

In this reduction we will continue and break each clause into a separate constraint leaving us with at most $qm2^q$ constraints in $\phi''$

Note that $\phi''$ is a 3SAT formula

Soundness holds since if $\phi$ were satisfiable $\phi'$ and $\phi''$ would also be satisfiable

Completeness holds since at least a $\epsilon$ fraction of the constraints were invalid even more would be invalid in $\phi''$ as we are creating equivalent but (polynomial) longer formulas.
As stated in the beginning of this talk, one of the major influences of PCP theorem is that an approximation algorithm for some NP problem is likely to not be a good approximation for another NP problem. Even though exact solutions of NP problems through Cook-Levin-Karp reductions are equivalent, this statement induces a potential hierarchy of difficulty presented by approximation algorithm solutions to NP problems. A very glaring example of this non-equivalence is evident in the approximation algorithms for Max Independent Set and Vertex Cover, as they are exact complements of one another.
Alt. Definition INDSET

The Largest Independent Set is the complement of the smallest vertex cover

Theorem (11.5)

There is some $\gamma < 1$ such that computing a $\gamma$-approximation to MIN-VERTEX-COVER is NP-Hard. For every $\rho < 1$, computing a $\rho$-approximation to INDSET is NP-Hard
Some Arithmetic with approximations

- VC = cardinality of the minimal vertex cover set
- IS = cardinality of the largest independent set
- $VC = n - IS$
- a $\rho$-approximation of independent set would be of size $\rho \cdot IS$
- and Min-Vertex-Cover would be of size $n - \rho \cdot IS$
- which yields an approximation ratio $\frac{n - \rho \cdot IS}{n - IS}$ for Min-Vertex-Cover

According to Theorem 1.15 (PCP Theorem), if $P \neq NP$, and using our $\frac{1}{2}$ approximation of Min-Vertex-Cover, then INDSET does not have a $\rho$-approximation algorithm.
Lemma 11.16

There exists a polynomial-time computable transformation $f$ from 3CNF formula to graphs such that for every 3CNF formula $\varphi$, $f(\varphi)$ is an $n$-vertex graph whose largest independent set has size $\text{val}(\varphi) \frac{n}{7}$

from this we can extract the following corollary

Corollary 11.17

if $P \neq NP$ then there are some constants $\rho < 1$, $\rho' < 1$ such that the problem INDSET cannot be $\rho$-approximated in polynomial time and MIN-VERTEX-COVER cannot be $\rho'$-approximated
Sketch of Proof of Corollary 11.17 Part I

1. use thm. 1.9 to show that any NP problem can be solved by MAX-3SAT

2. use the reduction lemma 1.16 on the 3CNF to reduce it to INDSET

3. compute a $\rho$-approximation of INDSET

this implies that there is a $\rho$-approximation of MAX-3SAT
therefor the $\rho$-approximation to INDSET is NP-Hard
using the arithmetic relating the set sizes of MIN-VERTEX-COVER and INDSET we get

4 $VC = n - val(\varphi) \frac{n}{7}$, where $val(\varphi)$ is the number of satisfied 3CNFs

5 if MIN-VERTEX-COVER has a $\rho'$-approximation where $\rho' = 6/(7 - \rho)$

6 this allows us to find a vertex cover of size $\frac{1}{\rho'}(n - \frac{n}{7})$ in the case $val(\varphi) = 1$

7 and the size is at most $n - \rho n/7$ which allows us to distinguish between $val(\varphi) = 1$ and $val(\varphi) < \rho$ which by thm. 1.9 is NP-hard
Definition

Graph Product - a binary operation on a graph such that

- the vertex sets of H is the product of the two graphs
- the vertices are connected in the resulting graph if in the original graphs they are connected, and are also connected to the vertices they form a product with.
Graph Product Example

1 * 6 = 9

L. Carraher and R. McGovern
PCP Theorem and Hardness of Approximation
Proof 1.15

- for any n-vertex graph G, define the graph $G^k$ to be a graph on $\binom{n}{k}$ vertices.
- $G^k$ will have Subsets of G of size k.
- Two subsets, $S_1$ and $S_2$ are adjacent if their union in $G^k$ is an independent set.
- The largest independent set in $G^k$ corresponds to the largest independent set in G, and has size $\binom{IS}{k}$
- If we use the graph produced by corollary 1.17 and take its k-wise product, then the size of the independent set differs by a factor $\rho^k$.
- this allows us to choose a k such that $\rho$ is smaller than any constant.
- the reduction is $n^k$ which is polynomial for any constant k
Theorem 11.19

\[
\text{NP} \subseteq \text{PCP}(\text{poly}(n), 1)
\]
Definition Walsh-Hadamard Code

Method for encoding bit strings of length \( n \) by linear functions in \( n \) variables over \( \text{GF}(2) \).

\[ \text{WH} : \{0, 1\}^* \rightarrow \{0, 1\}^* \]

maps a string \( u \in \{0, 1\}^n \) to the truth table of function \( x \mapsto u \circ x \),
where \( x, y \in \{0, 1\}^n \)

\[ u \circ x = \sum_{i=1}^{2n} x_i y_i \mod 2 \]

- the WH mapping can be implemented as a matrix transform following some permutations to the 1’s vector
- first the sequence is repeated across the column and rows
- next the sequence is negated on the diagonal
Walsh Code Transform

\[
\begin{pmatrix}
-1 \\
-1 & -1 \\
-1 & 1 \\
-1 & 1 & -1 & 1 \\
-1 & -1 & 1 & 1 \\
-1 & 1 & 1 & -1 \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
-1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\
-1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 \\
-1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
-1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 \\
-1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\
\end{pmatrix}
\]

... do this \( n \) times to get the walsh-transform \( WH \).
Random Subsum Principle

if $u \neq v$ then for $\frac{1}{2}$ the choices of $x$, $u \odot x \neq v \odot x$

- the Walsh-Hadamard Code is a systemic code - the original message is contained in the codeword
- $WH(x)$ is the evaluation of every linear function at $x$
- since any two linear functions disagree on exactly half the set of inputs, the fraction distance of the walsh-hadamard code is $1/2$.
- the walsh-hadamard code is a $[2^k, k, 2^{k-1}]$-code
because the Walsh-Hadamard codes are exactly the linear functions from \( \{0, 1\}^n \) to \( \{0, 1\} \) then we can test with:

\[
f(x + y) = f(x) + f(y)
\]

however this requires reading all \( 2^n \) values of \( f \).

A natural assumption for PCP is to test \( x,y \) at random and verify (11.3)

this fails however for a function \( f \) that is close to a linear function, due to testing on only a small subset of \( f \)'s components will not distinguish it from the linear function that it is close too

instead, propose a test that accepts if \( f \) is a linear function, and rejects if \( f \) is far from linear.
Definition 11.20

Let $\rho \in [0, 1]$. We say that $f, g : \{0, 1\}^n \rightarrow \{0, 1\}$ are $\rho$-close if $Pr_{x \in \{0, 1\}^n}[f(x) = g(x)] \geq \rho$. We say that if $f$ is $\rho$-close to a linear function if there exists a linear function $g$ such that $f$ and $g$ are $\rho$-close.

Theorem 11.21 Linearity Testing

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be such that $Pr_{x, y \in \{0, 1\}^n}[f(x + y) = f(x) + f(y)] \geq \rho$ for some $\rho > \frac{1}{2}$ then $f$ is $\rho$-close to a linear function.
$1 - \delta$ linearity test

for every $\delta \in (0, \frac{1}{2})$ we obtain a linearity test (11.3) that with probability $> \frac{1}{2}$ rejects every function that is not $(1 - \delta)$-close to a linear function. Repeat $O(1/\delta)$ times

Local Decoding Algorithm

- Choose $x' \in_R \{0, 1\}^n$
- Set $x'' = x + x'$
- Let $y' = f(x')$ and $y'' = f(x'')$
- output $y' + y''$
x' and x'' are uniformly distributed
by the union bound, with probability $1 - 2\delta$ we have $y' = f^*(x')$ and $y'' = f^*(x'')$, where $f^*$ is a linear function near $f$
by linearity of $f^*$: $f^*(x) = f(x' + x'') = f^*(x') + f^*(x'')$
which implies that with at least probability $1 - 2\delta$, $f^*(x) = y' + y''$
this is the local decoding of the Walsh-Hadamard Code as it corrects $f$, the bit corrupted codeword, to $f^*$ the linear function, with only a constant number of queries.
The proof for theorem 1.19 works by creating a Verifier for a NP complete language using an encoded version of the usual certificate.

To do this we will use the NP-Complete language QUADEQ.

Let $V$ be a $PCP(poly(n), 1)$ and has access to proof $\pi \in \{0, 1\}^{2^n + 2^{n^2}}$ which we interpret as a pair of functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and $g : \{0, 1\}^{n^2} \rightarrow \{0, 1\}$.

In a correct proof for QUADEQ $f$ will be the Walsh-Hadamard encoding for the satisfying assignment $u$ and $g$ will be the Walsh-Hadamard encoding for $u \otimes u$. 

Proof of Theorem 1.19

- Step 1: Verify that both $f$ and $g$ are linear functions
- Step 2: Verify $g$ encodes $u \otimes u$ where $u$ is the string encoded by $f$
- Step 3: Verify $g$ encodes a satisfying assignment
Step 1: Verify that both $f$ and $g$ are linear functions
This is done by random sampling to verify that $f$ and $g$ are 0.999 close to a linear function
Step 2: Verify $g$ encodes $u \otimes u$ where $u$ is the string encoded by $f$

- We do this by verifying that $f(r)f(r') = g(r \otimes r')$ ten times
- This will allow us to halt on an incorrect proof $1/4$ of the time on each of the ten calls giving us a certainty of $\geq 0.9$ that we will halt with an incorrect proof.
Proof of Theorem 1.19

- Step 3: Verify $g$ encodes a satisfying assignment
- This can be done easily by checking any particular equation is satisfied by $u$
- Finally to check that $u$ satisfies the entire system we have to make a query to $g$ for each equation which violates the principle to fix this we use the random subsum principle and create a new quadratic equation.
- Thus if $u$ does not satisfy even one equation in the original then with probability at least $1/2$ it will not satisfy this new equation.
References

-S. Arora, B. Barak, Computational Complexity a Modern Approach, Cambridge University Press 2009
-Guruswami O’Donnel, “CSE 533: The PCP Theorem“, 2005 Proof of the PCP Theorem
http://www.cs.washington.edu/education/courses/cse533/05au/