Notes on the PCP Theorem and Complexity of Approximations

These notes combine sections from a survey paper on hardness of approximation that I wrote in 2010 and notes for a graduate complexity theory from 2008. This material is not part of the syllabus of CS172.

1 Hardness of Approximation

We know that a number of important optimization problems are NP-hard to solve exactly. Today we begin the study of the complexity of finding *approximate* solutions.

There is a fundamental difficulty in proving hardness of approximation results. All the **NP**-completeness proofs for graph problems before 1990 can be essentially described as follows: we start from the computation of a generic non-deterministic Turing machine, then we encode its computation as a 3SAT formula, using the construction of Cook's theorem, and then we reduce 3SAT to the problem of interest (the reduction may be presented as a sequence of reductions involving several intermediate problems, but it can always be thought of as a direct reduction from 3SAT) by encoding variables and clauses of the formula as sub-graphs connected in a proper way. The computation of a Turing machine is very sensitive to small changes, and it seems impossible to generate an inapproximability gap starting from a fragile model and applying "local" reductions. The only inapproximability results that can be proved with such reductions are for problems that remain **NP**-hard even restricted to instances where the optimum is a small constant. For example, in the Metric Min k-Center problem it is **NP**-hard to decide whether the optimum has cost 1 or 2, and so no algorithm can have a performance ratio smaller than 2 unless $\mathbf{P} = \mathbf{NP}$ [9]. Similarly, in the Coloring problem it is **NP**-hard to decide wether the optimum has cost 3 or 4, and so no algorithm has performance ratio smaller than 4/3 unless $\mathbf{P} = \mathbf{NP}$, and Garey and Johnson [8] show that the gap can be "amplified" to k versus $2k - 4$ for constant k, ruling out also algorithms with performance ratio smaller than 2. Most interesting problems, however, become trivial when restricted to inputs where the optimum is a constant.

To prove more general inapproximability results it seemed necessary to first find a machine model for **NP** in which accepting computations would be "very far" from rejecting computations. Before such a model was discovered, an important piece of work on inapproximability was due to Papadimitriou and Yannakakis, who showed that, assuming that Max 3SAT does not have a PTAS,[1] then several other problems do not have a PTAS [14]. Berman and Schnitger [3] proved that if Max 2SAT does not have a PTAS then, for some $c > 0$, the Independent Set problem cannot be approximated within a factor n^c.

The modern study of inapproximability was made possible by the discovery, due to Feige et al. [6] in 1990, that probabilistic proof systems could give a robust model for **NP** that could be used to prove an inapproximability result for the Independent Set problem.[2] A year later, Arora et al. [2, 1] proved the PCP Theorem, a very strong characterization of **NP** in terms of proof systems, and showed how to use the PCP Theorem to prove that Max 3SAT does not have a PTAS. Using

[1] PTAS, short for *Polynomial Time Approximation Scheme*, is an algorithm that given an instance x and a parameter ϵ runs in time polynomial in n, but with an arbitrary dependency on ϵ, and returns a $(1 + \epsilon)$-approximate solution.

[2] Another, less influential, connection between probabilistic proof checking and inapproximability was discovered around the same time by Condon [5].

the reductions of [14] (and others [13, 4]), the PCP Theorem gave inapproximability results for several other problems.

2 Probabilistically Checkable Proofs

Probabilistically checkable proofs (PCPs) provide a "robust" characterization of the class **NP**. When we reduce a generic **NP** problem L to 3SAT using Cook's theorem, we give a way to transform an instance x into a 3CNF formula ϕ_x so that if $x \in L$ then ϕ_x is satisfiable, and if $x \notin L$ then ϕ_x is not satisfiable. Following the proof of Cook's theorem, however, we see that it is always easy (even when $x \notin L$) to construct an assignment that satisfies all the clauses of ϕ_x except one.

Using the PCP Theorem one can prove a stronger version of Cook's theorem, that states that, in the above reduction, if $x \in L$ then ϕ_x is satisfiable, and if $x \notin L$ then there is no assignment that satisfies even a $1 - \epsilon$ fraction of clauses of ϕ_x, where $\epsilon > 0$ is a fixed constant that does not depend on x. This immediately implies that Max 3SAT does not have a PTAS (unless $\mathbf{P} = \mathbf{NP}$), and that several other problems do not have a PTAS either (unless $\mathbf{P} = \mathbf{NP}$), using the reductions of Papadimitrious and Yannakakis [14] and others.

We define PCPs by considering a probabilistic modification of the definition of **NP**. We consider probabilistic polynomial time verifiers V that are given an input x and "oracle access" to a witness string w. We model the fact that V is a probabilistic algorithm by assuming that V, besides the input x and the witness w, takes an additional input R, that is a sequence of random bits. Then V performs a deterministic computation based on x, w and R. For fixed w and x, when we say "$V^w(x)$ accepts" we mean "the event that V accepts when given oracle access to witness w, input x, and a uniformly distributed random input R." When we refer to the "probability that $V^w(x)$ accepts," we take the probability over the choices of R.

We say that a verifier is $(r(n), q(n))$-restricted if, for every input x of length n and for every w, $V^w(x)$ makes at most $q(n)$ queries into w and uses at most $r(n)$ random bits.

We define the class $\mathbf{PCP}[r(n), q(n)]$ as follows. A language L is in $\mathbf{PCP}[r(n), q(n)]$ if there is an $(r(n), q(n))$-restricted verifier V such that if $x \in L$, then there is w such that $V^w(x)$ accepts with probability 1 and if $x \notin L$ then for every w the probability that $V^w(x)$ accepts is $\leq 1/2$.

We also consider the following more refined notation. We say that a language L is in $\mathbf{PCP}_{c(n),s(n)}[r(n), q(n)]$ if there is an $(r(n), q(n))$-restricted verifier V such that if $x \in L$, then there is w such that $V^w(x)$ accepts with probability at least $c(n)$, and if $x \notin L$ then for every w the probability that $V^w(x)$ accepts is at most $s(n)$. Of course, the definition makes sense only if $0 \leq s(n) < c(n) \leq 1$ for every n. The parameter $c(n)$ is called the *completeness* of the verifier and the parameter $s(n)$ is called the *soundness error*, or simply the *soundness* of the verifier.

Note that if $r(n) = O(\log n)$ then the proof-checking can be *derandomized*, that is, V can be simulated by a polynomial time deterministic verifier that simulates the computation of V on each of the $2^{r(n)} = n^{O(1)}$ possible random inputs and then computes the probability that $V^w(x)$ accepts, and then accepts if and only if this probability is one. It then follows that, for example, $\mathbf{PCP}[O(\log n), O(\log n)] \subseteq \mathbf{NP}$. The PCP Theorem shows a surprising converse.

Theorem 1 (PCP Theorem) $\mathbf{NP} = \mathbf{PCP}[O(\log n), O(1)]$.

The theorem was proved in [2, 1], motivated by a relation between PCP and approximation discovered in [6], and based on much previous work. In the notes for the past lectures we discussed the historical context

3 PCP and the Approximability of Constraint Satisfaction Problem

In the Max 3SAT problem we are given a 3CNF boolean formula, that is, a boolean formula in conjunctive normal form (AND-of-OR of literals, where a literal is either a variable or the negation of a variable) such that each term in the conjunction is the OR of at most three literals. The goal is to find an assignment that satisfies the largest possible number of terms.

In the Max qCSP problem, where q is a positive integer, we are given a system of boolean constraints defined over boolean variables such that every constraints involves at most q variables. The goal is to find an assignment that satisfies as many constraints as possible. Note that Max 3SAT is a special case of Max 3CSP. (View each clause as a constraint.)

Theorem 2 *The* **PCP** *Theorem implies that there is a constant q such that there is no a-approximate algorithm for Max qCSP with $a < 2$, unless* $\mathbf{P} = \mathbf{NP}$.

PROOF: Let $L \in \mathbf{PCP}[O(\log n), q]$ be an **NP**-complete problem, where q is a constant, and let V be the $(O(\log n), q)$-restricted verifier for L. We describe a reduction from L to Max qCSP.

Given an instance z of L, our plan is to construct a Max qCSP instance I_z with $m = |z|^{O(1)}$ constraints such that

$$\begin{array}{rcl} z \in L & \Rightarrow & I_z \text{ is satisfiable} \\ z \notin L & \Rightarrow & \mathrm{opt}_{\text{Max qCSP}}(I_z) \leq \frac{m}{2} \end{array} \tag{1}$$

Once (1) is proved, the theorem follows.

We enumerate all random inputs R for V. The length of each string is $r(|z|) = O(\log |z|)$, so the number of such strings is polynomial in $|z|$. For each R, V chooses q positions $i_1^R, \ldots, i_q^R$ and a Boolean function $f_R : \{0,1\}^q \to \{0,1\}$ and accepts iff $f_R(w_{i_1^R}, \ldots, w_{i_q^R}) = 1$.

We want to simulate the possible computation of the verifier (for different random inputs R and different witnesses w) as a Boolean formula. We introduce Boolean variables $x_1, \ldots, x_\ell$, where ℓ is the length of the witness w.

For every R we add the constraint $f_R(x_{i_1^R}, \ldots, x_{i_q^R}) = 1$. This completes the description of I_z.

Now, if $z \in L$, then there is a witness w that is accepted by V with probability 1. Consider the assignment $x_i \leftarrow w_i$, where w_i is the i-th bit of w: then such an assignment satisfies all the constraints of I_z.

If I_z has an assignment $x_i \leftarrow a_i$ that satisfies more than $m/2$ constraints, then the witness w defined as $w_i := a_i$ is accepted with probability more than $1/2$ by V, which implies that $z \in L$. So if $z \notin L$ then I_z has no assignment that satisfies more than $m/2$ constraints. □

Theorem 3 *The* **PCP** *Theorem implies that is an $\epsilon_1 > 0$ such that there is no polynomial time $(1+\epsilon_1)$-approximate algorithm for Max-3SAT, unless* $\mathbf{P} = \mathbf{NP}$.

PROOF: Given an instance I of Max qCSP (where q is the constant in the PCP Theorem) over variables $x_1, \ldots, x_n$ and with m constraints, we show how to construct an instance ϕ_I of Max 3SAT with m' clauses such that

$$\begin{array}{rcl} I \text{ is satisfiable} & \Rightarrow & \phi_I \text{ is satisfiable} \\ \mathrm{opt}_{\text{Max qCSP}}(I) \leq \frac{m}{2} & \Rightarrow & \mathrm{opt}_{\text{Max 3SAT}}(\phi_I) \leq (1-\epsilon_1)m' \end{array} \tag{2}$$

Once (2) is proved, the theorem follows.

For every constraint $f(x_{i_1}, \ldots, x_{i_q}) = 1$ in I, we first construct an equivalent qCNF of size $\leq 2^q$. Then we "convert" clauses of length q to clauses length 3, which can be done by introducing additional variables, as in the standard reduction from kSAT to 3SAT (for example $x_2 \vee x_{10} \vee x_{11} \vee x_{12}$ becomes $(x_2 \vee x_{10} \vee y_R) \wedge (\bar{y}_R \vee x_{11} \vee x_{12})$). Overall, this transformation creates a formula ϕ_I with at most $q2^q$ 3CNF clauses for each constraint in I, so the total number of clauses in ϕ_I is at most $q \cdot 2^q \cdot m$.

Let us now see the relation between the optimum of ϕ_z as an instance of Max 3SAT and the optimum of I as an instance of Max qCSP.

If I is satisfiable, then set the x variables in ϕ_I to the same values and set the auxiliary variables appropriately, then the assignment satisfies all clauses, and ϕ_I is satisfiable.

If every assignment of I contradicts at least half of the constraints, then consider an arbitrary assignment to the variables of ϕ; the restriction of the assigment to the x variables contradicts at least $m/2$ constraints of I, and so at least $m/2$ of the clauses of ϕ_I are also contradicted. The number $m' - m/2$ is at most $m'(1 - \epsilon_1)$ if we choose $\epsilon_1 \leq \frac{1}{2q2^q}$. □

Interestingly, the converse also holds: any gap-creating reduction from an **NP**-complete problem to Max qCSP implies that the **PCP** Theorem must be true.

Theorem 4 *If there is a reduction as in (1) for some problem L in* **NP***, then $L \in$* **PCP**$[O(\log n), q]$. *In particular, if L is* **NP***-complete then the* **PCP** *Theorem holds.*

PROOF: We describe how to construct a verifier for L. V on input z expects w to be a satisfying assignment for I_z. V picks a constraint of at random, and checks that the assignment $x_i \leftarrow w_i$ satisfies it. The number of random bits used by the verifier is $\log m = O(\log |z|)$. The number of bits of the witness that are read by the verifier is q.

$$\begin{aligned} z \in L &\Rightarrow I_z \text{ is satisfiable} \\ &\Rightarrow \exists w \text{ such that } V^w(z) \text{ always accept.} \end{aligned}$$

$$\begin{aligned} z \notin L &\Rightarrow \forall w \text{ a fraction } \tfrac{1}{2} \text{ of constraints of } I_z \text{ are unsatisfied by } w \\ &\Rightarrow \forall w \ V^w(z) \text{ rejects with probability } \geq \tfrac{1}{2} \end{aligned}$$

□

4 Irit Dinur's Proof of the PCP Theorem

In this section we outline the proof of the PCP theorem.

Theorem 5 $\mathbf{NP} \subseteq \mathbf{PCP}_{c=1, s=\frac{1}{2}}(O(\log(n)), O(1))$

To do this we will construct instances of constraint satisfaction problems (CSPs) for which it is hard to distinguish the case in which the CSP is satisfiable from the case in which every assignment contradicts a constant fraction of constraints.

We will work with the type of CSPs where each constraint has two variables, but where each variable can take on a non-boolean (but constant-size) range of values.

Definition 6 Max Cut
Input: variables $x_1, \ldots, x_n$ that range over Σ, a collection of binary constraints.
Goal: find an assignment that maximizes that number of satisfied constaints.

Definition 7 *If C is a CSP, we call $opt(Max\mathcal{C})$ the fraction of constraints which are satisfied by the optimal assignment.*

The following is the version of the PCP Theorem that we will prove.

Theorem 8 *There exists a Σ_0, a polynomial time reduction R, and a $\delta_0 > 0$ such that*

- *R is a reduction from 3-coloring to Max Cut$_0$.*
- *If G is 3-colorable, then $R(G)$ is satisfiable.*
- *If G is not 3-colorable, then opt(C) $\leq 1 - \delta_0$.*

This theorem implies the PCP theorem because given a graph G, we can define a valid proof to be a binary encoding of a solution to the constraint satisfaction problem $R(G)$. Given an alleged proof, the verifier randomly picks $O(\frac{1}{\delta_0})$ constraints to check, reads an assignment for the variables in such constraints from the proof, and accepts if and only if all constraints are satisfied.

The verifier uses $O(\log(n))$ random bits and reads $O(\frac{1}{\delta_0} \log |\Sigma_0|)$ bits of the proof. (We assume that the assignment to the n variables is encoded as a string of $n \log |\Sigma_0|$ bits.) If R works as in the theorem statement, then if G is three colorable, the CSP is satisfiable and there exists a valid proof that is accepted with probability 1. Furthermore, if G is not three colorable, then, for every alleged proof, a δ_0 fraction of the constraints in $R(G)$ will not be satisfied. Therefore, with probability at least $\frac{1}{2}$ the verifier will choose a constraints that is not satisfied, and thus reject.

Observe that 2-CSP-$\{a, b, c\}$ is at least as hard is 3-coloring because 3-coloring can be set up as a 2-CSP over a three-element range. We see from the theorem statement that

- $\text{opt}(G) = 1 \Rightarrow opt(R(G)) = 1$.
- $\text{opt}(G) \leq 1 - \frac{1}{|E|} \Rightarrow opt(R(G)) \leq 1 - \delta_0$.

The idea will be to create R by amplifying the fraction of unsatisfied constraints by a constant factor while only increasing the number of constraints by a linear amount and applying this amplification a logarithmic number of times. We can restate the theorem as follows:

Theorem 9 (restated) *There is δ_0, Σ_0, $|\Sigma_0| \geq 3$, and polynomial time R mapping inputs of Max Cut$_0$ to Max Cut$_0$ such that*

1. *# of constraints of $R(Max\mathcal{C}) = O(\#$of constraints of $R(Max\mathcal{C}))$.*
2. *$opt(Max\mathcal{C}) = 1 \Rightarrow opt(R(G)) = 1$.*
3. *$opt(Max\mathcal{C}) \leq 1 - \delta \Rightarrow opt(R(Max\mathcal{C})) \leq 1 - 2\delta$ if $\delta < \delta_0$.*

We prove this theorem using two lemmas. The first lemma will amplify the number of unsatisfiable constraints, but will also increase the range size. The second lemma will reduce the range size, but will decrease the number of unsatisfiable constraints.

Lemma 10 (Amplification) *$\forall \Sigma_0$, $\forall c$, there exists Σ and a poly-time R_1, mapping Max Cut$_0$ to Max Cut such that R satisfies 1) and 2) in Theorem 9 and $opt(Max\mathcal{C}) \leq 1 - \delta \Rightarrow opt(R_1(Max\mathcal{C})) \leq 1 - c\delta$ provided that $c \leq \delta_0$.*

Lemma 11 (Range Reduction) *$\exists \Sigma_0$, $\exists c_0$, such that for all Σ, there exists a poly-time R_2, mapping Max Cut to Max Cut$_0$ such that R satisfies 1) and 2) in Theorem 9 and $opt(Max\mathcal{C}) \leq 1-\delta \Rightarrow opt(R_2(Max\mathcal{C})) \leq 1 - \delta/c_0$.*

To get the theorem from these two lemmas, let $c = 2c_0$ in Lemma 10, then the composition $R_2(R_1(\cdot))$ solves the theorem because:

$$opt(Max\mathcal{C}) \leq 1 - \delta \Rightarrow opt(R_1(Max\mathcal{C})) \leq 1 - c\delta = 1 - 2c_0\delta \Rightarrow opt(R_2(R_1(Max\mathcal{C}))) \leq 1 - 2\delta$$

5 Basic Reductions

We have seen that the PCP Theorem is equivalent to the inapproximability of Max 3SAT and other constraint satisfaction problems. In this section we will see several reductions that prove inapproximability results for other problems.

5.1 Max 3SAT with Bounded Occurrences

We begin with a reduction from the Max E3SAT problem on general instances to the restriction of Max E3SAT to instances in which every variable occurrs only in a bounded number of clauses. The latter problem will be a useful starting point for other reductions.

For the reduction we will need *expander graphs* of the following type.

Definition 12 (Expander Graph) *An undirected graph $G = (V, E)$ is a 1-expander if, for every subset $S \subseteq V$, $|S| \leq |V|/2$, the number of edges $e(S, V - S)$ having one endpoint in S and one in $V - S$ is at least $|S|$.*

For our purposes, it will be acceptable for the expander graph to have multiple edges. It is easy to prove the existence of constant-degree 1-expanders using the probabilistic method. Polynomial-time constructible 1-expanders of constant degree can be derived from [7], and, with a smaller degree, from [12]. Let d be a constant for which degree-d 1-expanders can be constructed in polynomial time. ($d = 14$ suffices using the construction of [12].)

Let now ϕ be an instance of 3SAT with n variables $x_1, \ldots, x_n$ and m clauses. For each variable x_i, let occ_i be the number of *occurrences* of x_i, that is, the number of clauses that involve the literal x_i or the literal $\bar{x}_i$. We write $x_i \in C_j$ if the variable x_i occurs in clause C_j. Notice that $\sum_{i=1}^{n} occ_i = 3m$. For each i, construct a 1-expander graph $G_i = (V_i, E_i)$ where V_i has occ_i vertices, one for each occurrence of x_i in ϕ. We denote the vertices of V_i as pairs $[i, j]$ such that x_i occurrs in C_j. Each of these graphs has constant degree d.

We define a new instance ψ of Max E3SAT with $N = 3m$ variables $Y = \{y_{i,j}\}_{i \in [n], x_i \in C_j}$, one for each occurrence of each variable in ϕ. For each clause of ϕ we put an equivalent clause in ψ. That is, if $C_j = (x_a \vee x_b \vee x_c)$ is a clause in ϕ, then $(y_{a,j} \vee y_{b,j} \vee y_{c,j})$ is a clause in ψ. We call these clauses the *primary clauses* of ψ. Note that each variable of ψ occurs only in one primary clause.

To complete the construction of ψ, for every variable x_i in ϕ, and for every edge $([i, j], [i, j'])$ in the graph G_i, we add the clauses $(y_{i,j} \vee \bar{y}_{i',j})$ and $(\bar{y}_{i,j} \vee y_{i',j})$ to ψ. We call these clauses the *consistency clauses* of ψ. Notice that if $y_{i,j} = y_{i',j}$ then both consistency clauses are satisfied, while if $y_{i,j} \neq y_{i',j}$ then one of the two consistency clauses is contradicted.

This completes the construction of ψ. By construction, every variable occurrs in at most $2d+1$ clauses of ψ, and ψ has $M = m + 3dm$ clauses.

We now claim that the cost of an optimum solution in ψ is determined by the cost of an optimum solution in ϕ and, furthermore, that a good approximation algorithm applied to ψ returns a good approximation for ϕ. We prove the claim in two steps.

Claim 13 *If there is an assignment for ϕ that satisfies $m-k$ clauses, then there is an assignment for ψ that satisfies $\geq M-k$ clauses.*

PROOF: This part of the proof is simple: take the assignment for ϕ and then for every variable $y_{i,j}$ of ψ give to it the value that the assignment gives to x_i. This assignment satisfies all the consistency clauses and all but k of the remaining clauses. □

Claim 14 *If there is an assignment for ψ that leaves k clauses not satisfied, then there is an assignment for ϕ that leaves $\leq k$ clauses not satisfied.*

PROOF: This is the interesting part of the proof. Let $a_{i,j}$ be the value assigned to $y_{i,j}$. We first "round" the assignment so that all the consistency clauses are satisfied. This is done by defining an assignment b_i, where, for every i, the value b_i is taken to be the majority value of $a_{i,j}$ over all j such that $x_i \in C_j$, and we assign the value b_i to all the variables $y_{i,j}$. The assignment b_i satisfies all the consistency clauses, but it is possible that it contradicts some primary clauses that were satisfied by $a_{i,j}$. We claim that, overall, the b_i assignment satisfies at least as many clauses as the $a_{i,j}$ assignment. Indeed, for each i, if b_i differs from the $a_{i,j}$ for, say, t values of j, then there can be at most t primary clauses that were satisfied by $a_{i,j}$ but are contradicted by b_i. On the other hand, because of the consistency clauses being laid out as the edges of a 1-expander graph, at least t consistency clauses are contradicted by the $a_{i,j}$ assignment for that value of i alone, and so, the b_i assignment can be no worse.

We conclude that b_i assignment contradicts no more clauses of ψ than are contradicted by $a_{i,j}$, that is, no more than k clauses. When we apply b_i as an assignment for ϕ, we see that b_i contradicts at most k clauses of ϕ. □

In conclusion:

- If ϕ is satisfiable then ψ is satisfiable;
- If every assignment contradicts at least an ϵ fraction of the clauses of ϕ, then every assignment contradicts at least an $\epsilon/(1+3d)$ fraction of the clauses of ψ.

Theorem 15 *There are constants d and ϵ_2 and a polynomial time computable reduction from 3SAT to Max 3SAT-d such that if ϕ is satisfiable then $f(\phi)$ is satisfiable, and if ϕ is not satisfiable then the optimum of $f(\phi)$ is less than $1-\epsilon_2$ times the number of clauses. In particular, if there is an approximation algorithm for Max 3SAT-d with performance ratio better than $(1-\epsilon_2)$, then $\mathbf{P}=\mathbf{NP}$.*

5.2 Vertex Cover and Independent Set

In an undirected graph $G=(V,E)$ a vertex cover is a set $C \subseteq V$ such that for every edge $(u,v) \in E$ we have either $u \in C$ or $v \in C$, possibly both. An independent set is a set $S \subseteq V$ such that for every two vertices $u, v \in S$ we have $(u,v) \notin E$. It is easy to see that a set C is a vertex cover in G if and only if $V-C$ is an independent set. It then follows that the problem of finding a minimum size vertex cover is the same as the problem of finding a maximum size independent set. From the point of view of approximation, however, the two problems are not equivalent: the Vertex Cover

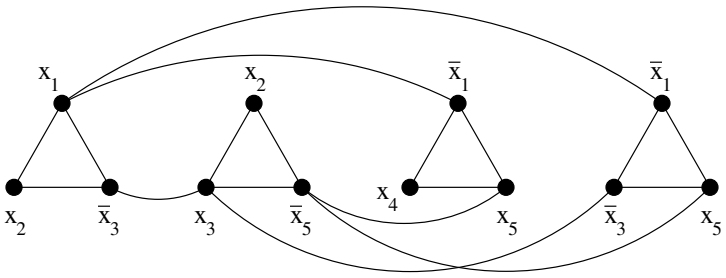

Figure 1: Graph construction corresponding to the 3CNF formula $\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_2 \vee x_3 \vee \bar{x}_5) \wedge (\bar{x}_1 \vee x_4 \vee x_5) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee x_5)$.

problem has a 2-approximate algorithm (but, as we see below, it has no PTAS unless $\mathbf{P} = \mathbf{NP}$), while the Independent Set problem has no constant-factor approximation unless $\mathbf{P} = \mathbf{NP}$.

We give a reduction from Max E3SAT to Independent Set. The reduction will also prove intractability of Vertex Cover. If we start from an instance of Max E3SAT-d we will get a bounded degree graph, but the reduction works in any case. The reduction appeared in [14], and it is similar to the original proof of **NP**-completeness of Vertex Cover and Independent Set [11].

Starting from an instance ϕ of E3SAT with n variables and m clauses, we constuct a graph with $3m$ vertices; the graph has a vertex $v_{i,j}$ for every occurrence of a variable x_i in a clause C_j. For each clause C_j, the three vertices corresponding to the three literals in the clause are joined by edges, and form a triangle (we call such edges *clause edges*). Furthermore, if a variable x_i occurrs positively in a clause C_j and negated in a clause $C_{j'}$, then there is an edge between the vertices $v_{i,j}$ and $v_{i,j'}$ (we call such edges *consistency edges*). Let us call this graph G_ϕ. See Figure 1 for an example of this construction.

Note that if every variable occurrs in at most d clauses then the graph has degree at most $d+2$.

Claim 16 *There is an independent set of size $\geq t$ in G_ϕ if and only if there is an assignment that satisfies $\geq t$ clauses in ϕ.*

PROOF: Suppose we have an assignment a_i that satisfies t clauses. For each clause C_j, let us pick a vertex $v_{i,j}$ that corresponds to a literal of C_j satisfied by a_i. We claim that the set of picked vertices is an independent set in G_ϕ. To prove the claim, we note that we picked at most one vertex from each triangle, so that we do not violate any clause edge, and we picked vertices consistent with the assignment, so that we could not violate any consistency edge.

For the other direction, suppose we have an independent set with t vertices. The vertices must come from t different triangles, corresponding to t different clauses. We claim that we can satisfy all such clauses. We do so by setting an assingment so that x_i takes a value consistent with the vertices $v_{i,j}$ in the independent set, if any. Since consistency edges cannot be violated, this is a well defined assignment, and it satisfies t clauses. □

If we combine this reduction with Theorem 15, we get the following result.

Theorem 17 *There is a polynomial time computable function mapping instances ϕ of 3SAT into graphs G_ϕ of maximum degree $d+2$ such that if ϕ is satisfiable then G_ϕ has an independent set of size at least $N/3$ (and a vertex over of size at most $2N/3$, where N is the number of vertices, and if ϕ is not satisfiable then every independent set in G_ϕ has size at most $N \cdot (1-\epsilon_2)/3$, and every vertex cover has size at least $N \cdot (2+\epsilon_2)/3$. In particular, if there is an approximation algorithm*

for Independent Set in degree-$(d+2)$ graphs with performance ratio better than $1/(1-\epsilon_2)$, or if there is an approximation algorithm for Vertex Cover in degree-$(d+2)$ graphs with performance ratio better than $1+\epsilon_2/2$, then $\mathbf{P} = \mathbf{NP}$.

5.3 Steiner Tree

In the Steiner tree problem we are given a graph $G = (V, E)$, with weights on the edges, and a subset $C \subseteq V$ of vertices. We want to find the tree of minimum cost that contains all the vertices of C. This problem is different from the minimum spanning tree problem because the tree is not required to contain the vertices in $V - C$, although it may contain some of them if this is convenient. An interesting special cases (called *Metric Steiner Tree*) arises when E is the complete graph and the weights are a metric. (That is, they satisfy the triangle inequality.) Without this restriction, it is not hard to show that the problem cannot be approximated within any constant.

We describe a reduction from the Vertex Cover problem in bounded degree graphs to the Steiner Tree problem. The reduction is due to Bern and Plassmann [4].

We start from a graph $G = (V, E)$, and we assume that G is connected.[3] We define a new graph G' that has $|V| + |E|$ vertices, that is, a vertex $[v]$ for each vertex v of G and a vertex $[u, v]$ for each edge (u, v) of G.

The distances in G' are defined as follows:

- For every edge $(u, v) \in E$, the vertices $[u]$ and $[u, v]$ are at distance one, and so are the vertices $[v]$ and $[u, v]$.
- Any two vertices $[u]$, $[v]$ are at distance one.
- All other pairs of vertices are at distance two.

We let C be the set of vertices $\{[u, v] : (u, v) \in E\}$. This completes the description of the instance of the Steiner Tree problem. Notice that, since all the distances are either one or two, they satisfy the triangle inequality, and so the reduction always produces an instance of Metric Steiner Tree.

Claim 18 *If there is a vertex cover in G with k vertices, then there is a Steiner tree in G' of cost $m + k - 1$.*

PROOF: Let S be the vertex cover. Consider the vertices $\{[v] : v \in S\}$ and the vertices $\{[u, v] : (u, v) \in E\}$, and consider the weight-one edges between them. We have described a connected sub-graph of G', because every vertex in $\{[u] : u \in S\}$ is connected to every other vertex in the same set, and every vertex $[u, v]$ is connected to a vertex in $\{[u] : u \in S\}$. Let us take any spanning tree of this subgraph. It has $m + k - 1$ edges of weight one, and so it is of cost $m + k - 1$, and it is a feasible solution to the Steiner Tree problem. □

Claim 19 *If there is a Steiner tree in G' of cost $\leq m + k$, then there is a vertex cover in G with k vertices.*

[3] We proved inapproximability of Vertex Cover without guaranteeing a connected graph. Clearly, if we have an approximation algorithm that works only in connected graphs, we can make it work on general graphs with same factor. It follows that any inapproximability for general graphs implies inapproximability of connected graphs with the same factor.

PROOF: Let T be a feasible Steiner tree. We first modify the tree so that it has no edge of cost 2. We repeatedly apply the following steps.

- If there is an edge of cost 2 between a vertex $[w]$ and a vertex $[u, v]$, we remove it and add the two edges $([w], [u])$ and $([u], [u, v])$ of cost 1.
- If there is an edge of cost 2 between a vertex $[u, v]$ and a vertex $[v, w]$, we remove it and add the two edges $([u, v], [v])$ and $([v], [v, w])$ of cost 1.
- Finally, and this case is more interesting, if there is an edge of cost 2 between the vertices $[u, v]$ and $[w, z]$, we remove the edge, and then we look at the two connected components into which T has been broken. Some vertices $[u, v]$ are in one component, and some vertices are in the other. This corresponds to a partition of the edges of G into two subsets. Since G is connected, we see that there must be two edges on different sides of the partition that share an endpoint. Let these edges be (u, v) and (v, w) then we can reconnect T by adding the edges $([u, v], [v])$ and $([v], [v, w])$.

We repeat the above steps until no edges of cost two remain. This process will not increase the cost, and will return a connected graph. We can obtain a tree by removing edges, and improving the cost, if necessary.

The final tree has only edges of weight one, and it has a cost $\leq m + k - 1$, so it follows that it spans $\leq m + k$ vertices. The m vertices $\{[u, v] : (u, v) \in E\}$ must be in the tree, so the tree has $\leq k$ vertices $[v]$. Let S be the set of such vertices. We claim that this is a vertex cover for G. Indeed, for every edge (u, v) in G, the vertex $[u, v]$ is connected to v_0 in the tree using only edges of weight one, which means that either $[u]$ or $[v]$ is in the tree, and that either u or v is in S. □

If we combine the reduction with the results of Theorem 17, we prove the following theorem.

Theorem 20 *There is a constant ϵ_3 such that if there is a polynomial time $(1 + \epsilon_3)$-approximate algorithm for Metric Steiner Tree then $\mathbf{P} = \mathbf{NP}$.*

5.4 More About Independent Set

In this section we describe a direct reduction from PCP to the Independent Set problem. This reduction is due to Feige et al. [6].

Let L be **NP**-complete, and V be a verifier showing that $L \in \mathbf{PCP}_{c,s}[q(n), r(n)]$. For an input x, let us consider all possible computations of $V^w(x)$ over all possible proofs w; a complete description of a computation of V is given by a specification of the randomness used by V, the list of queries made by V into the proof, and the list of answers. Indeed, for a fixed input x, each query is determined by x, the randomness, and the previous answers, so that it is enough to specify the randomness and the answers in order to completely specify a computation. We call such a description a *configuration*. Note that the total number of configuration is at most $2^{r(n)} \cdot 2^{q(n)}$, where n is the length of x.

Consider now the graph G_x that has a vertex for each *accepting* configuration of V on input x, and has an edge between two configurations c, c' if c and c' are *inconsistent*, that is, if c and c' specify a query to the same location and two different answers to that query. We make the following claims.

Claim 21 *If $x \in L$, then G_x has an independent set of size $\geq c \cdot 2^{r(n)}$.*

PROOF: If $x \in L$, then there is a proof w such that $V^w(x)$ accepts with probability at least c, that is, there is a proof w such that there are at least $c \cdot 2^{r(n)}$ random inputs that make $V^w(x)$ accept. This implies that there are at least $c \cdot 2^{r(n)}$ mutually consistent configurations in the graph G_x, and they form an independent set. □

Claim 22 *If $x \notin L$, then every independent set of G_x has size $\leq s \cdot 2^{r(n)}$.*

PROOF: We prove the contrapositive: we assume that there is an independent set in G_x of size $\geq s \cdot 2^{r(n)}$, and we show that this implies $x \in L$. Define a witness w as follows: for every configuration in the independent set, fix the bits in w queried in the configuration according to the answers in the configurations. Set the bits of w not queried in any configuration in the independent set arbitrarily, for example set them all to zero. The $s \cdot 2^{r(n)}$ configurations in the independent set correspond to as many different random strings. When $V^w(x)$ picks any such random string, it accepts, and so $V^w(x)$ accepts with probability at least s, implying $x \in L$. □

It follows that if there is a ρ-approximate algorithm for the independent set problem, then every problem in $\mathbf{PCP}_{c,s}[r(n), q(n)]$ can be solved in time $\text{poly}(n, 2^{r(n)+q(n)})$, provided $c/s < \rho$.

From the PCP Theorem we immediately get that there cannot be a ρ-approximate algorithm for the independent set problem with $\rho < 2$ unless $\mathbf{P} = \mathbf{NP}$, but we can do better.

Suppose that V is a $(O(\log n), O(1))$-restricted verifier for an **NP**-complete problem, and that V has soundness $1/2$ and completeness 1. Define a new verifier V' that performs two independent repetitions of the computation of V, and that accepts if and only if both repetitions accept. Then V' has clearly soundness $1/4$ and completeness 1, and it is still $(O(\log n), O(1))$-restricted, thus showing that even an approximation better than 4 is infeasible. If we repeat V a constant number of times, rather than twice, we can rule out any constant factor approximation for the independent set problem.

In general, a verifier that makes $k(n)$ repetitions shows that $L \in \mathbf{PCP}_{1,1/2^{k(n)}}[O(k(n) \cdot \log n), O(k(n))]$, and the reduction to Independent Set produces graphs that have $2^{O(k(n) \cdot \log n)}$ vertices and for which $2^{k(n)}$-approximate algorithms are infeasible. If we let $k(n) = \log n$, then the graph has size $N = 2^{O((\log n)^2)}$ and the infeasible ratio is n, which is $2^{\Omega(\sqrt{\log N})}$. So, if we have an algorithm that on graphs with N vertices runs in polynomial time and has an approximation ratio $2^{o(\sqrt{\log N})}$, then we have an $O(n^{O(\log n)})$ algorithm to solve 3SAT, and $\mathbf{NP} \subseteq \mathbf{QP}$. More generally, by setting $k(n) = (\log n)^{O(1)}$, we can show that if there is an $\epsilon > 0$ such that Independent Set can be approximated within a factor $2^{O((\log n)^{1-\epsilon})}$ then $\mathbf{NP} \subseteq \mathbf{QP}$.

Finally, using random walks in expander graphs as in [10], it is possible to use the **PCP** theorem to show that, for every $k(n)$, $\mathbf{NP} = \mathbf{PCP}_{1,1/2^{k(n)}}[O(k(n) + \log n), O(k(n))]$. If we choose $k(n) = \log n$, then, in the reduction, we have a graph of size $2^{O(k(n)+\log n)} = n^{O(1)}$ for which an approximation ratio of n is infeasible. This shows the following result.

Theorem 23 *There is a constant $c > 1$ such that if there is a polynomial time n^c-approximate algorithm for Independent Set then $\mathbf{P} = \mathbf{NP}$.*

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. Preliminary version in *Proc. of FOCS'92.*

[2] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. Preliminary version in *Proc. of FOCS'92*.

[3] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation*, 96:77–94, 1992. Preliminary version in *Proc. of STACS'89*.

[4] M. Bern and P. Plassmann. The Steiner tree problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[5] A. Condon. The complexity of the max-word problem and the power of one-way interactive proof systems. *Computational Complexity*, 3:292–305, 1993. Preliminary version in *Proc. of STACS91*.

[6] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. Preliminary version in *Proc. of FOCS91*.

[7] O. Gabber and Z. Galil. Explicit construction of linear sized superconcentrators. *Journal of Computer and System Sciences*, 22:407–425, 1981.

[8] M.R. Garey and D.S. Johnson. The complexity of near-optimal graph coloring. *Journal of the ACM*, 23:43–49, 1976.

[9] D.S. Hochbaum and D.B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[10] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[11] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[12] Alexander Lubotzky, R. Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988.

[13] C.H. Papadimitriou and M. Yannakakis. The travelling salesman problem with distances one and two. *Mathematics of Operations Research*, 18:1–11, 1993.

[14] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991. Preliminary version in *Proc. of STOC'88*.