The Tale of the PCP Theorem

How the search for the limits of computing led to the discovery of the unexpected power of proofs

Dana Moshkovitz, MIT

1. Introduction

In the 1970's, the landscape of algorithms research changed dramatically. It was discovered that an efficient algorithm for any one of numerous innocent-looking algorithmic problems would imply something that sounds too fantastic to be true: an algorithm that given a provable mathematical statement, efficiently finds the shortest proof for the statement. To get some estimate for how fantastic the latter is, such an algorithm, if exists, is currently worth at least $6,000,000 – promised by the Clay Institute to whoever solves six central math problems; $1 million dollars per problem. The algorithm would have efficiently solved all six problems. Amusingly, one of the six problems is to determine whether such an algorithm exists. This problem is known as the P vs. NP problem.

The innocent-looking algorithmic problems mentioned above, on the other hand, look like they could have easily had an efficient algorithm -- they are incredibly similar to problems for which we *do* know efficient algorithms. One example is Max-Cut: find the best way to separate the nodes of a network into two, so there are as many links as possible between the two parts. If you replace "as many" with "as little", you get Min-Cut, whose flow-based algorithm is studied by all Computer Science undergrads to this day. An efficient algorithm for Max-Cut, on the other hand, would imply the $6,000,000 worth algorithm, or, put in technical terms, Max-Cut is "NP-hard".

2. Approximation

A few years passed, and researchers observed something interesting: while they indeed did not come up with efficient algorithms for problems like Max-Cut, they could sometimes prove that certain algorithms will not do much worse than the optimum for the problems. For instance, here is a way to generate a Max-Cut partition such that the number of links between the two parts is at least half of the maximum (in technical terms this is a "½-approximation"): pick the partition at random. That is, for each node flip a coin. If the coin shows "heads", put the node in the first part, and if coin shows "tails", put the node in the second part. The probability that the two endpoints of a link belong to different parts is exactly half. Hence, on average, half of all links will connect nodes belonging to different parts. As it turns out, there is a better algorithm for Max-Cut, one that finds a partition in which the fraction of links between different parts is about 0.878 times the optimum (i.e., a ≈0.878-approximation). This algorithm was discovered by Michel Goemans and David Williamson in the 1990's and we will not describe it here. That algorithm has not been improved upon for the last two decades.

While Max-Cut has an approximation to within a constant factor, for other problems such approximations were not found; sometimes the best approximation depended on the size of the input, so as the input size grew bigger, the approximation worsened. In sharp contrast, some problems could be approximated arbitrarily well (to within 0.9999..9 for any number of 9's). This left researchers wondering whether they were way off with the other problems – could there exist better approximation algorithms that had not been discovered? The salvation came from an unexpected source.

3. Hardness of Approximation

By the early 1990's researchers realized how to show NP-hardness of approximation problems. This explained the little success algorithmists had with improving their approximations for some problems.

The conceptual path that led to this breakthrough was long and intertwined. It started a decade earlier with works of Goldwasser, Micali and Rackoff on zero-knowledge proofs, and a work of Babai on interactive proofs. Back then, no one foresaw these could have any implications to approximation algorithms, but this is what ended up happening.

Both works suggested models for proving and checking proofs that were different from the standard model of writing down a proof, and checking it step by step. The new models viewed theorem proving as a discussion between a prover and a verifier. The verifier may ask the prover questions, and the questions may be randomized. If the verifier's impression of the validity of the proof is correct with probability, say, 0.99, they said, this is good enough.

Those models and related models then became the subject of study and classification that showed that they were surprisingly powerful. This research eventually created the know-how that led to hardness of approximation.

4. Robust Proofs and the PCP Theorem

- So what implies hardness of approximation? "Merely" the following theorem that no one who has ever written or checked a mathematical proof would consider possible, but nevertheless turns out to be true:

"*Every* mathematical proof can be written in a format that can be checked probabilistically by reading only **two** statements (no matter how many statements there are in the proof)."

A standard proof of size n is a sequence of statements, ordered 1,…,n. Every statement is implied by some of the previous statements and the premises. The last statement is what we want to prove. Importantly, the only way to make sure the last statement is correct using the standard proof is to follow the reasoning, statement by statement. If there's a mistake in any one of the implications, the statement may be wrong.

In contrast, in the new proof format, many statements can each imply the same statement. For example, it is possible that the statement in the 5'th position determines the statement in the 10'th position, but also that the statement in the 16'th position determines the statement in the 10'th position. This structure makes the following "robustness" property possible: even if only a small fraction of all implications in the new format holds, one can still deduce the entire reasoning of the original proof.

This robustness property makes probabilistic checking possible: To check the proof, one picks a random pair of positions such that the statement in the first position is supposed to imply the statement in the second position. One then tests that this is indeed the case. For a correct proof, the first statement implies the second. On the other hand, if the bottom line of the proof is false, i.e., there is no reasoning that implies it, then almost all implications must not hold. Hence, the probability that the first statement implies the second is low.

The theorem became known as the PCP Theorem, where PCP stands for "Probabilistically Checkable Proofs".

Incidentally, as some readers may recognize, PCP also stands for "Phenylcyclohexyl Piperidine", a recreational dissociative drug. According to Muli Safra, a researcher who participated in proving one of the first versions of the PCP theorem, the name commemorates an incident in which a police force mistakenly suspected that Safra and his friends were running a PCP lab. Safra thought that the least they could do after being falsely accused is to actually run a PCP lab (albeit of a different nature).

5. How the PCP Theorem Implies Hardness of Approximation

To complete the picture, let us explain how PCP is relevant to hardness of approximation.[1]

[1] The explanation will be self-contained, but might be difficult to follow without any background in NP-hardness.

Suppose that you want to prove hardness of 0.95-approximation for Max-Cut. We claim that you will be done if you show the following: given a network, it is NP-hard to distinguish between the case where there is a partition of the nodes into two sets such that an X-fraction of all links is between different parts, and the case where in every partition of the nodes into two parts, at most Y-fraction of the links is between different parts, assuming Y/X=0.95. Why will you be done? Because if you could approximate the maximal fraction of links between two parts to within 0.95, you could distinguish between the two cases. The distinguishing problem is often called a *gap problem* because of the gap between the fractions X and Y.

Next you want to show that the gap problem is as hard as testing whether a proof of size n exists for a given math statement, since this would show that 0.95-approximating Max-Cut is NP-hard. This is done by means of a *reduction*: showing an efficient way of translating the testing problem to the gap problem so that an efficient algorithm for the gap problem implies an efficient algorithm for the testing problem, which in turn means that the gap problem is at least hard as the testing problem. Concretely, you translate a math statement and the problem "is there a proof of size at most n for the math statement, or does every proof require size larger than n?" into a network and the problem "is there a way to partition the nodes of the network into two parts such that the fraction of links between the different parts is at least X, or does the best partition have at most a Y-fraction of the links?".

Assuming your reduction is correct, then a correct proof of size n for the math statement translates into a partition of the nodes in the network that has at least an X –fraction of the links between the two parts. The new format of the proof is thus an assignment of each one of the nodes to either "part 1" or "part 2". Interestingly, to check the new proof, it is enough to pick a random link and check whether its endpoints are in different parts (i.e., whether the first endpoint being in one part implies that the second endpoint is in the other part). A correct proof guarantees that the probability of correct implication is at least X, while an incorrect proof guarantees that the probability of implication is at most Y. Overall, the reduction from "is there a proof of size n?" to the gap problem is essentially equivalent to a transformation of a proof to a probabilistically checkable format. In other words, it is a PCP Theorem.

This PCP Theorem is slightly different from the PCP theorem we described before: *X*, the probability of rejecting a correct proof, is not 1. Moreover, while the probability *Y* of accepting a proof for an incorrect statement is lower than *X*, it may not be much lower than *X*. These are artifacts of checking a proof using "Max-Cut tests". If we change the type of tests we allow, there are methods to get the probability of rejecting a correct proof to 0, and the probability of accepting a proof for an incorrect statement to close to 0.

6. The Proof of the PCP Theorem

The PCP theorem has a reputation for being difficult and complicated to prove, which, in the author's opinion, is unjustified. We are going to outline the proof in this section. The purpose of the outline is to get a feel for the proof, rather than full details. Thus, we will often skip technical details, and omit in-depth explanations of some of the components.

The proof of the PCP theorem repeatedly decreases the number of statements one needs to read from the proof, so it eventually goes down from n (all statements) to 2. Below we describe only one decrease, which can then be applied recursively. The recursion, called "composition" in PCP jargon, is tricky, and, while we nowadays know of elegant ways to do it, they still involve significant loss in parameters. We skip the details of composition in this presentation.

We assume a proof format and a probabilistic verifier that reads q statements from the proof. We describe a new proof format and a probabilistic verifier that reads much fewer than q statements. The new format is an encoding of the existing proof by what is known in the theory of error correcting codes as a *Reed-Muller code*. This code is an *algebraic* code - its definition involves finite fields and multivariate polynomials. This may seem surprising to readers who encounter such constructions for the first time, but indeed there are numerous examples of mathematical arguments that have nothing to do with polynomials in which polynomials turn out to be very useful.

Consider a finite field *F* and let *H* be a small set of finite field elements so that $n=|H^m|$ for some parameter *m*. The Reed-Muller code interpolates an *m*-variate

low degree polynomial over a finite field from the existing proof, interpreting the bits of the existing proof as the evaluations of the polynomial on the product subset H^m. The new format also contains some auxiliary data as described below. Low degree multivariate polynomials serve two purposes: the first- they form an error correcting code, which means that to move from one polynomial to another one has to change the values of most of the points in F^m; the second- they have a recursive structure: the restriction of a polynomial to fewer variables, by substituting values to the other variables, is again a low degree polynomial.

One can show (and we'll skip the details here) that, without loss of generality, in order to verify the proof it suffices to check whether the sum of a certain low degree polynomial $p(x_1,\ldots,x_m)$ over H^m is 0 (p is not exactly the Reed-Muller polynomial, but it depends on it and on the given verifier). Note that a-priori it seems like one needs to read $n=|H^m|$ positions from the proof to check the sum, just like one would need to read n statements from the proof to check it. We are going to show that reading only about $|H|\cdot m$ positions suffices! We will do that by adding auxiliary data to the new proof format.

The auxiliary data is the "partial sums" of the sum $\Sigma_{x1,\ldots xm \text{ in } H} p(x_1,\ldots,x_m)$ we want to check. A partial sum fixes the first i coordinates, and sums only over the remaining coordinates:

$$S_i(x_1,\ldots,x_i) := \Sigma_{xi+1,\ldots xm \text{ in } H} p(x_1,\ldots,x_i,\ldots,x_m), \text{ for } i=0,\ldots,m.$$

In the new format of the proof, we will store $S_i(x_1,\ldots,x_i)$ for all $i=0,\ldots,m$ and all $x_1,\ldots,x_i$ in F ("the table of S_i").

By definition, the partial sums are related to each other and to p. The exact equations they satisfy are these:

1. $S_0() = \Sigma_{x1,\ldots xm \text{ in } H} p(x_1,\ldots,x_m)$
2. $S_{i-1}(x_1,\ldots,x_{i-1}) = \Sigma_{xi \text{ in } H} S_i(x_1,\ldots,x_i)$ for $i=1,\ldots,m+1$
3. $S_m(x_1,\ldots,x_m) = p(x_1,\ldots,x_m)$

Next we describe the verifier. The verifier first checks that the tables of the S_i's are (approximately) low degree polynomials. This can be done by reading evaluations on a random line, and is known as "low degree testing". As we

mentioned above, polynomials of low degree have a remarkable property: two different low degree polynomials differ on almost all the points. Thus, the equality in 3 can be checked probabilistically by checking it on a *single* random point, and the m+1 equalities in 2 can be checked probabilistically by checking each on |H|+1 points. By equality 1, this allows verification of the sum over all n points. The total number of queries is (m+1)(|H|+1)+2, instead of $|H|^m$. The large saving in the number of queries was made possible by the pre-computing done by the partial sums. Importantly, this pre-computing can be checked using the recursive structure and the code property of the multivariate polynomials.

In 2005, Dinur found a different, beautiful, proof for the PCP theorem based on expanders. This proof only gives high error probability, say 0.999, as opposed to the very low error probability that can be derived from the proof we sketched above. That error probability can then be decreased by other means.

7. Optimal Hardness Results

While the PCP theorem yielded the first hardness of approximation results in the early 1990's, it took until around 1997 before such hardness results could match the best approximations known for various problems.

The path from the PCP theorem to optimal hardness results is typically comprised of the following steps:

1. *Soundness amplification* - in case the PCP theorem has a high error probability, one first lowers this probability. This is usually done by the *parallel repetition lemma* of Raz (1994), whose proof is based on information-theoretic ideas.

2. Composition of the PCP with the "long code", a specialized construction due to Bellare-Goldreich-Sudan (1995), which allows one to tailor the PCP to the specific approximation problem in question.

3. Analysis of the long code by means of Fourier analysis, as was first done by Håstad (1997). The analysis relies on the low error of the PCP.

8. The Unique Games Conjecture

So how hard is it to approximate Max-Cut? Håstad showed that Max-Cut is NP-hard to approximate better than 16/17 ≈ 0.941. Is there a better algorithm than the Goemans-Williamson ≈ 0.878-approximation? We do not know. However, as often is the case in theoretical computer science, we can make a "Ramsfeldian" assertion about our state of knowledge: we might know how to know if there is not.

In 2002 Khot put forward the *Unique Games Conjecture* (UGC), which postulates the existence of a certain special PCP to be explained below. The UGC, if true, settles the approximability of Max-Cut, and determines that the Goemans-Williamson algorithm is optimal [Khot-Kindler-Mossel-O'Donnell, 2004]. If true, the UGC also settles a host of other problems, including NP-hard problems concerning covering, partitioning and ordering. More than that, the conjecture points to a certain algorithmic technique, called "semi-definite programming", that would yield optimal approximation algorithms [Raghavendra, 2008].

What is the Unique Games Conjecture? The "uniqueness" refers to checking two-sided implication instead of a one-sided implication: the statement in the first position implies the statement in the second position, and vice versa, the statement in the second position implies the statement in the first position. It may seem unlikely to check any proof using only if-and-only-if's. However, the unique games conjecture states that it is possible -- at least as long as one uses an interesting trick: a tiny fraction of the if-and-only-if's are allowed to be false even in a correct proof. This makes unique verification plausible, and the conjecture is that arbitrarily low error probability can be achieved by such a verification process.

The PCP obtained from Max-Cut is unique: if there is a partition of the nodes into two parts, such that at least X fraction of the links are between the two parts, then for all links, except for at most 1-X fraction, one endpoint of the link is in one part if and only if the other endpoint is in the other part. The unique games

conjecture states that very low error probability is achievable if one considers a generalization of Max-Cut in which there are many possible parts, not just two.

9. More Open Problems

Even if the Unique Games Conjecture is resolved, other problems will remain open. For instance, for the traveling salesman problem, in which a salesman wants to minimize the cost of tour through given destinations, we do not even have a candidate approach for proving tight hardness of approximation results.

Various intriguing open problems remain regarding the original PCP theorem. One is minimizing the error probability. At present, the lowest error probability we know for the theorem (for the "two query projection" version that was introduced in this article) is logarithmically small in the size of the proof [M-Raz 2008]. One could a-priori hope for polynomially small error (the "projection games conjecture"/"sliding scale conjecture"). Another open problem is minimizing the blow-up introduced by the probabilistically checkable format: by research that was done in the 2000's, we know that the size of the proof format does not have to increase by more than a sub-linear factor to make the format probabilistically checkable. Will an increase by a constant factor suffice?

A different question is whether PCP can be applied in practice. The small blow-ups, low error probabilities and conceptually simpler constructions we know today might lead to a practical "PCP technology". When would it be useful? This requires an asymmetric setting, in which the prover is willing to invest some extra work in preparing the proof, and the verifier is required to be extremely efficient. The verifier may not trust the prover, and must be certain that the prover committed to a proof, e.g., by writing it down. One can envision settings in which such conditions hold - a server and weak agents for example.

10. Bibliographic Remarks and Further Reading

The first proof of the PCP theorem by Arora-Safra and Arora-Lund-Motwani-Sudan-Szegedy appeared in 1992. Weaker PCP theorems, with a larger number of

queries were known before, by works of Lund-Fortnow-Karloff-Nisan, Babai-Fortnow-Lund and Babai-Fortnow-Levein-Szegedy. The connection between local checking of proofs and hardness of approximation first emerged in a 1991 work by Feige-Goldwasser-Lovász-Safra-Szegedy. Papadimitriou and Yannakakis presented a family of candidate hard-to-approximate problems. After the proof of the PCP theorem, Lund and Yannakakis provided further hardness results. References to further developments were made in the body of the article.

There are various lecture notes and surveys about the PCP theorem. In particular, the author gives a course on PCP, and lecture notes are available from her website.