18.218 Topics in Combinatorics Spring 2021 – Lecture 15

Dor Minzer

Our next topic will be applications of results from analysis of Boolean functions in the field of hardness of approximation. Our goal will be to show that assuming a conjecture in complexity theory known as the Unique-Games Conjecture, the best known efficient approximation algorithms for the Max-Cut problem, as well as for the Vertex-Cover problem, achieve essentially the best approximation ratio possible by any polynomial time algorithm.

1 A brief introduction to complexity theory

Complexity theory studies the boundary between feasible and infeasible for computational problems. Usually, when we say "feasible", we mean that a problem is solvable in polynomial time in the size of the input.

1.1 Examples

We consider a few examples here:

Reachability in graphs. Consider the problem in which one is given as input a graph $G = (V, E)$ and two vertices $s, t \in V$. The goal is to decide whether the graph contains a path from s to t. The size of the input here is $O(|V| + |E|)$, and this problem can be solved in linear time (for example by either the BFS or DFS algorithms).

Matrix invertibility. Suppose we are given a matrix $A \in \mathbb{F}_2^{n \times n}$ as input, and we wish to decide whether A is invertible or not. This problem can be solved using Gaussian elimination, which takes $O(n^3)$ time, which is polynomial in the input size (which is $O(n^2)$).

Bipartiteness. Suppose we are given a graph $G = (V, E)$, and we wish to decide if G is bipartite or not. I.e., we want to decide whether there is a partition $V = A \cup B$ of the vertices into two sets, such that all edges go across them. At first sight, it's not clear whether one can solve this problem efficiently or not; one attempt is to go over all possible bipartitions A, B, but this takes time $O(2^n)$, which is exponential. Upon further inspection though, there is an alternative easy algorithm: start with $A = B = \emptyset$, and take any vertex $v \in V$ arbitrarily and put it in A. Now put all the neighbours of v in B; now in each step, take a vertex from either A or B, and put all of its neighbours in the other part. If we ever reach a contradiction – we say the graph is not bipartite, otherwise we find a bipartition this way. Furthermore, the running time of this algorithm is $O(|V| + |E|)$.

3-SAT. A 3-CNF formula is a formula of the form $\phi = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee \overline{x_{17}} \vee x_{11}) \wedge \ldots \wedge (x_1 \vee x_8 \vee x_{11})$. I.e., we have n-variables $x_1, \ldots, x_n$, and ϕ is an AND of m-clauses, each clause containing an OR over 3 literals (a literal is a variable of its negation). The input in the 3-SAT problem is to decide, given a 3-CNF formula ϕ, whether it is satisfiable or not. I.e., whether there is an assignment of $0, 1$ values to the variables that satisfies all of the clauses of ϕ.

Clearly, there is an easy algorithm that runs in time $O(m2^n)$ that solves the 3-SAT problem; slightly faster algorithms with running time $2^{\gamma n}$ where $\gamma < 1$ are known, but the best known algorithms for this problem still have exponential running time. It is widely believed that there are no polynomial time algorithms for this problem (this is equivalent to the P$\neq$ NP conjecture), but currently it is not even known how to rule out the existence of algorithms with running time $O(n \log^{10} n)$.

Maximum Cut. The Max-Cut problem is a generalization of the bipartiteness problem above. Given a graph $G = (V, E)$, one wants to find the bipartition $V = (A, B)$ that maximizes the number of edges that cross this bipartition. We note that the formulation of the problem we have here differs from the formulation of the problems above; the above problems were phrased as "decision problems" wherein the output is either 0 or 1 depending on whether the input satisfies a certain condition or not. Here, we have phrased the problem as an optimization problem, and one can state a decision analog of this problem.[1] In this case, it is also not known how to solve this problem efficiently, and it is again suspected that it is in fact impossible.

Minimum Vertex-Cover. Given a graph $G = (V, E)$, a vertex cover $C \subseteq V$ is a set of vertices that touches each edge $e \in E$, i.e. it contains at least one of its endpoints. The goal in the Minimum Vertex-Cover problem is, given a graph $G = (V, E)$, find the smallest vertex-cover in it. We do not know how to solve this problem efficiently, and again suspect that it is impossible.

1.2 NP-hardness

The above examples already give a taste of a theme going on in complexity theory. In order to evidence the feasibility of a computational problem, "all" one to do is demonstrate an efficient algorithm for it. How can one prove that there is *no* efficient algorithm for a problem though? After all, general computation may be very complex and do a whole bunch of clever manipulations that are not immediately evident. You can even think of the problem of proving mathematical theorems this way; given a statement, finding a proof for it may be a highly non-trivial task. Mostly for this reason, there are no results of this form (this is known as "proving lower bounds" in computational complexity), and we only know how to argue about very limited models of computation.

Still though, we would like to gain confidence to support our belief that there is no efficient algorithm exists for problems such as 3-SAT, Maximum-Cut, or Minimum Vertex-Cover. The main way complexity theorist do things along these lines is by the means of *reductions*, which we shortly explain. But before that, let's state a theorem that reductions are able to establish.

Theorem 1.1. *Suppose there exists an efficient algorithm for the Max-Cut problem (or the Minimum Vertex-Cover problem). Then there exists an efficient algorithm for the 3-SAT problem.*

In fact, the reverse direction also holds. If there is an efficient algorithm for 3-SAT, then there is also an efficient algorithm for the Max-Cut problem as well as for the Minimum Vertex-Cover. In other words, as far as polynomial algorithms are concerned, these problems are in the same basket. Loosely speaking,

[1] We will not give a comprehensive treatment of these distinctions in this course.

this points out that the "root cause" we were unable to find an efficient algorithm for each of these problems is the same. This is the beginning of the theory of NP-completeness and NP-hardness which started out in the 70's. Formally, we say a problem is NP-hard, if assuming we have an efficient algorithm for it, we can construct an efficient algorithm for the 3-SAT problem; in this language, the above theorem simply states that Max-Cut and Minimum Vertex-Cover are NP-hard.

1.3 Coping with NP-hardness

The fact that NP-hard problems are something people face very often, both in theory and in practice, has led researchers to consider ways to handle such problems. For example, can one find sub-classes of instances of 3-SAT that occur in some area (say, formal verification of programs), and show that for this class of instances there is an efficient algorithm? This is one way to cope with NP-hardness that people do look at which we will not discuss further.

Another option is to relax our notion of what it means to solve a problem. Namely, suppose we have a 3-CNF formula which is satisfiable, but instead of finding an assignment that satisfies all of the clauses, we manage to find an assignment satisfying at least 99% of the clauses. In many cases, this will be almost as good, so it makes sense to ask whether this is possible or not. This leads us to discuss *approximation problems*.

2 Approximation problems

We move on to discuss approximation versions of some of the problems we have presented thus far.

2.1 Approximately solving 3-SAT

Suppose we are given a 3-CNF formula ϕ and we are promised there is a satisfying assignment. What is the best assignment we can find efficiently? The most obvious thing to try is a random assignment: for each i, take x_i as uniform bit. Note that the probability a clause is satisfied is at least $7/8$, and so in expectation a random assignment satisfies at least $7/8m$ clauses. This observation can easily be turned into an efficient algorithm (we will not do it explicitly here). Further note that we managed to satisfy $7/8m$ clauses without even using the fact that ϕ was satisfiable. Can we do any better?

2.2 Approximately solving Vertex-Cover

Suppose we have a graph $G = (V, E)$, and we are promised that there is a vertex-cover of size γn. What is the smallest vertex cover we can find efficiently?

Claim 2.1. *There is an efficient algorithm that finds a vertex-cover of size at most $2\gamma n$.*

Proof. The algorithm starts with $C = \emptyset$, and $E' = E$. At each step, the algorithm takes some edge $e \in E'$, adds both of its endpoints to C, and then removes from E' all edges that contain at least one of these vertices as endpoints.

Clearly this is an efficient algorithm, and next we argue that $|C| \leqslant 2\gamma n$ in the end of the algorithm. Let $\tilde{C}$ be any other vertex cover of G. Note that at each time we add vertices to C, $\tilde{C}$ must contain at least one of these vertices (otherwise it wouldn't cover the edge we inspected), so we get that $|C| \leqslant 2\ |\tilde{C}|$. □

Thus, we can find a 2-approximation for Vertex-Cover. Can we do any better?

2.3 Approximately solving Max-Cut

Suppose we have a graph $G = (V, E)$, and we are promised that there is a cut of size $(1 - \varepsilon)m$. What is the largest cut we can find efficiently? Choosing a bipartition $V = A \cup B$ uniformly, one can show that the expected number of edges that go across it is $\frac{1}{2}m$, and this again can be turned into a $\frac{1}{2}$-approximation algorithm. Can we do any better?

3 The PCP theorem

In order to answer questions regarding the feasibility and infeasibility for approximation problems, one has to extend the theory of NP-hardness to the realm of approximation problems. This is the type of problems that are handled in the field of hardness of approximation, which essentially started with a result known as the PCP theorem. The theorem has several equivalent formulations, one of which is the following formulation using the notion of gap problems we introduce next.

Let $0 \leqslant s < c \leqslant 1$. The gap-3SAT$[c, s]$ problem is the promise problem wherein one is given a 3SAT instance ϕ, which is promised to belong in one of the following cases:

1. **YES case**: there is an assignment to ϕ satisfying at least c fraction of the clauses in ϕ.
2. **NO case**: no assignment satisfies more than s fraction of the clauses in ϕ.

The goal in the problem is distinguish which one of these cases does the given instance belong to.

Intuitively, the reason to consider such problems is that they give a convenient formalism that captures approximation; if one cannot even distinguish between formulas that are c-satisfiable and s-satisfiable, then one can certainly not approximate the 3-SAT problem within factor s/c.

The notion of gap problems can be applied to general optimization problem, and we will indeed use it in order to study the complexity of approximation problems.

Theorem 3.1 (The PCP Theorem)**.** *There exists $s < 1$, such that gap-3SAT$[1, s]$ is NP-hard.*

We will not prove this theorem in this course (it may well take us several lectures to do so). Instead, we will outline some applications of this result, and give some hint as to how analysis of Boolean functions enters the picture in such problems.

3.1 Implications of the PCP theorem

Starting out with the PCP theorem, one can prove a host of hardness of approximation results. For example, one can show that the $7/8$ approximation algorithm we had for the 3SAT problem is essentially tight:

Theorem 3.2 (Håstad 97')**.** *For all $\varepsilon > 0$, gap-3SAT$\left[1, \frac{7}{8} + \varepsilon\right]$ is NP-hard.*

One can also prove some hardness results for the other two problems we've discussed, though in this case the hardness result don't quite match the algorithms we've seen.

Theorem 3.3 (Håstad, Trevisan-Sorkin-Sudan-Williamson 00')**.** *It is NP-hard to approximate the Max-Cut problem within factor $\frac{16}{17}$.*

Despite much effort, this NP-hardness result still stands as best to date.

Theorem 3.4 (Dinur-Safra 02')**.** *It is NP-hard to approximate the Minimum Vertex-Cover problem within factor 1.36.*

Here, better NP-hardness results have been recently proved, and currently stand at $\sqrt{2} - o(1)$.

3.2 The Unique-Games Conjecture

So how does one make progress on these problems in order to determine the approximation ratio wherein these problems become computationally infeasible? To remedy this situation, Khot observed in a 2002 that if a "dream version" of the PCP theorem was proved, then it is conceivable that one could make progress on these problems. Towards this end, Khot formulated a conjecture called the Unique-Games Conjecture, asserting that a dream version of the PCP theorem holds. To state this conjecture, we first have to define the Unique-Games problem.

Definition 3.5. *An instance of Unique-Games, denoted by Ψ, is composed of a bipartite, bi-regular graph $G = (V = L \cup R, E)$, a finite alphabet Σ, and a collection of constraints $\Phi = (\phi_e)_{e \in E}$ one for each edge. Each one the constraint ϕ_e is a 1-to-1 map, $\phi_e \colon \Sigma \to \Sigma$.*

For an edge e, the constraint ϕ_e defines a collection of tuples which are deemed as satisfactory assignments to the endpoints of the edge, which is $\{(\sigma, \phi_e(\sigma)) \mid \sigma \in \Sigma\}$.

The goal in Unique-Games is to find an assignment $A \colon V \to \Sigma$ that satisfies as many of the constraints ϕ_e. The value of a Unique-Games instance, denoted by $\mathsf{val}(\Psi)$, is defined to be

$$\max_{A \colon V \to \Sigma} \frac{\#\{e \mid A \text{ satisfies } \phi_e\}}{|E|},$$

i.e. the maximum fraction of constraints that can be satisfied by any assignment.

It is easy to see that given a Unique-Games instance Ψ promised to have $\mathsf{val}(\Psi) = 1$, one can efficiently find an assignment that satisfies all of the constraints in Ψ (how?). What about if we are only given the weaker promise that $\mathsf{val}(\Psi) \geqslant 0.99$? The previous algorithm clearly breaks, and Khot conjectured that in this case it is hard to find a really good assignment.

Conjecture 3.6 (The Unique-Games Conjecture)**.** *For all $\varepsilon, \delta > 0$, there exists $k \in \mathbb{N}$ such that given a Unique-Games instance Ψ, it is NP-hard to distinguish between:*

1. **YES case***: $\mathsf{val}(\Psi) \geqslant 1 - \varepsilon$.*
2. **NO case***: $\mathsf{val}(\Psi) \leqslant \delta$.*

In other words, gap-UniqueGames$_k[1 - \varepsilon, \delta]$ is NP-hard.

Khot initially showed that if (a variant of this) conjecture is true, then Vertex-Cover is hard to approximate within factor $\sqrt{2} - o(1)$. Shortly after proposing this conjecture, researchers have realized that using Unique-Games as a starting problem, one can show tight inapproximability result for a wide range of computational problems. Furthermore, such results are intimately related to analysis of Boolean functions, and often reduce to proving some results about Boolean functions. In fact, some of the results that we've seen in this course were directly motivated by such applications.

Indeed, in this course we will see 2 implications of the Unique-Games Conjecture, namely to the problem of determining the complexity of approximating the Minimum Vertex-Cover and the Maximum Cut.

18.218 Topics in Combinatorics: Analysis of Boolean Functions

Spring 2021