6.890: Algorithmic Lower Bounds: Fun With Hardness Proofs Fall 2014

Lecture 12 Scribe Notes

Prof. Erik Demaine

1 Recap

For the past two classes, we've been covering inapproximability, some of its reduction styles, and some related classes of problems. In particular, we focused on contraint satisfiability problems (CSP) in the last lecture.

2 Overview

Today, we will cover gap reductions as this final part of our series on inapproximability, based on the c-gap problem, which converts optimization problems to decision problems. This is useful because if the c-gap problem is NP-hard, then so is c-approximating the original problem.

Gap reductions come in three flavors:

1. Gap-producing reductions,
2. Gap-amplification reductions, and
3. Gap-preserving reductions.

We covered each of these in class today, but focused mainly on gap-producing and gap-preserving reductions. We will then use c-gap to obtain optimal lower bounds on the 3SAT family of problems.

Finally, we will touch upon the Unique Games Conjecture, a point of great debate in modern times.

3 c-gap problem

We define the c-gap problem as a decision problem where we would like to distinguish between the upper and lower bounds of our optimal value OPT. In particular, we will define the c-gap problem for maximum-optimization problems and minimum-optimization problems.

3.1 Minimum-optimization definition of c-gap

For maximum-optimization problems, we would like to distinguish between two cases for OPT (where $c > 1$):

1. YES instance: $OPT \leq k$, and

2. NO instance: $OPT > c \cdot k$.

Note that c is not necessariy constant – it can be a function $c(n)$, for n is the size of the input. We will continue to refer to it as just c for the rest of lecture.

3.2 Maximum-optimization definition of c-gap

For maximum-optimization problems, we would like to distinguish between two cases for OPT:

1. YES instance: $OPT \geq k; c > 1$, and
2. NO instance: $OPT < \frac{k}{c}; c > 1$ (or, $OPT < c \cdot k; c < 1$, similar to what we saw in inapproximability).

Again, c is not necessarily a constant.

3.3 Assumptions

For c-gap problems, we will assume (i.e. we are promised) that OPT is within the two cases that we wish to distinguish.

3.4 Hardness

One notable result of c-gaps is: given this gap between your YES and NO instances, if your problem is *still* hard, then c-approximating the original problem must be hard.

That is to say, if the c-gap version of a problem is hard, then so is c-approximating it.

This result is stronger than what we have been using so far.

4 (a,b)-gap SAT/CSP

Our first application of c-gap is to (a, b)-gap satisfiability (SAT) and constraint satisfiability problems (CSP). We will indiscriminately consider both at the same time, as the application to both is roughly the same.

As a reminder, our goal is to maximally satisfy the clauses we are given, so our conversion of SAT and CSP to a maximum-optimization c-gap problem is as follows:

Given some clauses, we would like to distinguish between

1. $OPT \geq b \cdot (\# \text{ of clauses})$, which is our YES instance, and
2. $OPT < a \cdot (\# \text{ of clauses})$, which is our NO instance.

Rearranging terms to match our definition of the c-gap problem, we let $c = \frac{b}{a}$, and record the instances as:

1. YES: $OPT \geq \#$ of clauses, and
2. NO: $OPT < \frac{\#\text{ of clauses}}{c}$

Since we are introducing a gap here, note that $b > a$, so that the gap does exist, thus $\frac{b}{a} > 1$, matching our contraints on c for this problem. In fact, we typically set b to be 1 for convenience (i.e. satisfying everything) and let a be the greatest permitted fraction of non-satisfiable clauses.

5 Gap reductions

We will use the convention of reducing *from A to B*. Now, there are three ways to apply gaps in reductions:

1. Create a gap of some size, e.g. 1 (gap production)
2. Given a starting gap, we multiply that gap to be a larger gap (gap amplification)
3. Given a starting gap, we preserve the magnitude of that gap (gap preservation)

Today, we will start at gap production, touch upon gap amplification, and spend some more time at gap preservation.

5.1 Gap-producing reductions

Given some input, our optimal output has one of two possibilities:

1. YES instance: $OPT = k$, or
2. NO instance: there is a bound, based on the type of problem
    - Max: $OPT < \frac{k}{c}$, or
    - Min: $OPT > c \cdot k$.

Notice that we have taken the definition of the c-gap problem and replaced the YES instance with a fixed target for our optimal value, and thus produced a gap in our optimization problem that we're reducing from.

5.1.1 Simple example: Tetris

In Tetris, we can create a gap with $c = n^{1-\varepsilon}$ for some $\varepsilon > 0$. If we let OPT be the number of lines that can be cleared, then this gap is generated with the YES instance as solving the puzzle correctly, and the NO instance as a maximum-optimization for how many lines can be cleared. Letting ε be our tolerance for error, as we increase ε, our gap widens, and our related maximum bound for the NO instance decreases.

5.1.2 Another simple example: Nonmetric TSP

The nonmetric traveling salesman problem is, given a graph containing edges of weight 0 or 1, we would like find a path that visits each node exactly once, with minimum cost. We can reduce from Hamiltonian cycle for this as such: given a graph $G(V, E)$, we construct a new graph $G'(V, E')$, which is complete, and assign weights to each edge as follows: for each $e \in E'$,

1. If $e \in E$, assign e to have a weight of 0, otherwise,
2. If $e \notin E$, assign e to have a weight of 1.

We then apply the nonmetric traveling salesman problem to this graph, and note that a Hamiltonian cycle exists iff the nonmetric traveling salesman oracle can produce a path of weight 0 from G' (since that means we traversed only edges that existed in the original problem).

Alternatively, we could give each edge a weight of 1 or infinity (instead of 0 and 1, respectively), and check to see if the cost of the path is equal to $|V|$. This is preferred if we do not allow any edges of weight 0 in our graph.

This results in a huge gap (either $\frac{|E|}{0}$ if we're using 0's and 1's, or $\frac{\infty}{|E|}$ if we're using 1's and ∞'s).

5.1.3 PCP

Formally, denoted as $PCP(O(\log(n)), O(1))$, PCP stands for Probabilistic Checkable Proof. These proofs are created from the existence of an $O(1)$-time algorithm that can take a "certificate" generated by a solution to a problem, and with high probability check whether the solution is correct. It is assumed that the algorithm is given $O(\log n)$ bits of randomness, which it can read in $O(1)$.

When the solution is indeed correct, the algorithm must state that solution is valid. However, when the solution is invalid, the algorithm is allowed mistakenly state that the solution is valid within a fixed probability which is less than one.

If one were to run this algorithm a repeated number of times ($\log \frac{1}{\varepsilon}$) it is possible reduce the error to ε.

5.1.4 $(< 1, 1)$-gap 3SAT $\in$ PCP

We can now imagine using PCP on a 3SAT problem. In this case, the corresponding certificate would be the state of all variables. One suitable algorithm would be to take a random clause and verify that it is true. This would always return true if the solution is valid, and return a false positive with a probability proportional to the number of clauses.

$\Pr(wrong) < \frac{1}{gap}$

This has led to the PCP-theorem, which states that if $(< 1, 1)$-gap 3SAT is NP-hard, NP=PCP. Conversely, if 3SAT $\in$ PCP, then $(< 1, 1)$-gap 3SAT is NP-hard.

5.2 Gap-amplification reductions

As it turns out, a PCP algorithm runs in $O(1)$ time, and any $O(1)$-time algorithm can be written as a $O(1)$-size CNF formula. This is how we will create a gap 3SAT problem.

Based on the certification algorithm above, we can take a conjunction over $n^{O(1)}$ random choices of clauses. Then, given whether or not the proposed assignments do satisfy the whole CNF formula, we know something about the behavior of this certification. So, if the assignments should result in:

- YES, then the certification will always also result in YES. This is obvious because a valid assignment would never cause a clause to be false, so any random selection of clauses would always be satisfied in this case. However, if the assignments should result in:
- NO, then the certification will sometimes result in YES, and other times result in NO, depending on whether any the unsatisfied clause(s) are in our random selection. If $\Omega(1)$ fraction of the terms is false, then $\Omega(1)$ fraction of the clauses are also false, so we can apply a gap based on what fraction of our clauses are allowed to be false, which calibrates how many of the clauses we should check.

Thus, we have amplified 3SAT by starting from one gap (true vs false) and amplifying it into a larger gap (how many clauses are true vs false).

5.3 Gap-preserving reduction

Given an instance x of A, we would like to convert it to an instance x' of B using a function f such that $|x| = n$, and $|x'| = |n'|$.

Then, we would like the functions $k(n), k'(n'), c(n), andc'(n')$ to satisfy the following conditions:

1. $c(n) \geq 1, c'(n') \geq 1$ (this is part of our gap problem definitions)
2. For min problems:
   (a) $OPT_A(x) \leq k \implies OPT_B(x') \leq k'$, and
   (b) $OPT_A(x) \geq c \cdot k \implies OPT_B(x') \geq c' \cdot k'$

3. For max problems:
   (a) $OPT_A(x) \geq k \implies OPT_B(x') \geq k'$, and
   (b) $OPT_A(x) \leq \frac{k}{c} \implies OPT_B(x') \leq \frac{k'}{c'}$

Note that this relation is transitive: if $A \to B$ and $B \to C$, then $A \to C$. Furthermore, if $c' > c$, then because gap amplification, instead of preservation.

We will now look at some problems that utilize gap preservation.

5.3.1 MAX E3-X(N)OR-SAT

The goal here is to satisfy as many clauses of the form

$$x_i \oplus NOT(x_b) \oplus x_k$$

This is related to solving a system of linear equations of three terms.

The PCP version of this problem is $(\frac{1}{2} + \varepsilon, 1 - \varepsilon)$-gap, and is NP-hard, $forall \varepsilon > 0$.

We can then calculate the inapproximability by taking the quotient of the min and max bounds:

$$\frac{\frac{1}{2} + \varepsilon}{1 - \varepsilon} = \frac{1}{2} - \varepsilon$$

This leads to the conclusion that MAX E3-X(N)OR-SAT is $\left(\frac{1}{2} - \varepsilon\right)$-inapproximable.

However, a $\frac{1}{2}$-approximation does exist: we use uniform random assignment, and the Pr{correct parity for the equation} $= \frac{1}{2}$, since there are only 2 possibilities for each term.

5.3.2 MAX-E3SAT

In this problem, each clause has 3 distinct literals and want to maximize number of clauses satisfied. We will use a L-reduction from MAX-E3-X(N)OR-SAT:

$x_i \oplus x_j \oplus x_k = 1$ is represented as $(x_i \vee x_j \vee x_k) \wedge (\neg x_i \vee \neg x_j \vee x_k) \wedge (\neg x_i \vee x_j \vee \neg x_k) \wedge (x_i \vee \neg x_j \vee \neg x_k)$

$x_i \oplus x_j \oplus x_k = 0$ is represented as $(\neg x_i \vee \neg x_j \vee \neg x_k) \wedge (\neg x_i \vee x_j \vee x_k) \wedge (x_i \vee \neg x_j \vee x_k) \wedge (x_i \vee x_j \vee \neg x_k)$

Between these two cases, all possible permutations of the three literal's values have been considered. An easy way to remember this is: if the XOR evaluates to 1, then the representation has an even number of negations; if the XOR evaluates to 0, then the representation has an odd number of negations. Both of these come almost directly from how we defined the MAX-E3-X(N)OR-SAT problem.

In this representation, if the original XOR clause was satisfied, then its representative conjunction of four CNF clauses must all be satisfied; however, if the original XOR clause was not satisfied, then our representation will have exactly 3 of the clauses satisfied. This last point is not entirely trivial to see, but the logic is as follows: if the XOR clause was not satisfied, then our representation should evaluate to false, which means at least one of the clauses must be false. If we iterate through each of the four clauses, making each of them false one at a time, we then see that, by making one clause false, it sets the value of that variable, causing the remaining three clauses to always evaluate true. Thus, either 4 clauses are satisfied (if the original XOR is satisfied), or exactly 3 clauses are satisfied (if the original XOR is not satisfied).

This particular property allows us to preserve the additive error $\beta = 1$ that counts how many mistakes we make. Then, using a 2-approximation, we can see that

$$OPT_{E3SAT} = OPT_{E3-X(N)OR-SAT} + 3 \cdot \#\text{equations} \leq 7 \cdot OPT_{E3-X(N)OR-SAT},$$

since we have just shown that $OPT_{E3-X(N)OR-SAT}$ carries a value of 4 in our OPT_{E3SAT} space. Thus, $\alpha = 7$. Letting $\varepsilon = \frac{1}{2}$, we find that there is no $\left(1 - \frac{1}{2}\right)$-approximation for MAX-E3-X(N)OR-SAT, which translates to $\left(1 - \frac{\frac{1}{2}}{\alpha\beta}\right)$-approximation for MAX-E3SAT. Since $\alpha = 7, \beta = 1$, this is equivalent to saying that MAX-E3SAT has no $\left(1 - \frac{1}{2 \cdot 7 \cdot 1}\right) = \frac{13}{14}$-approximation.

However, we can do better! Using gaps, we can argue that:

1. YES instances have $\geq (1-\varepsilon) \cdot m$ of the equations satisfied, which translates to $\geq (1-\varepsilon) \cdot 4m + \varepsilon \cdot 3m$ of the clauses satisfied (remember which number correlates to satisfcation, and which one correlates to no satisfaction), and this simplies to simply $\geq \frac{(4-\varepsilon)m \text{ clauses satisfied}}{4m \text{ total clauses}}$.
2. NO instances have $< \left(\frac{1}{2} + \varepsilon\right) \cdot m$ of the equations satisfied, so applying the same substitutions, we see that this means that $< \left(\frac{1}{2} + \varepsilon\right) \cdot 4m + \left(1 - \frac{1}{2} - \varepsilon\right) \cdot 3m$ clauses are satisfied, which simplifies to $< \frac{\left(\frac{7}{2} + \varepsilon\right) m \text{ clauses satisfied}}{4m \text{ total clauses}}$.

This leads to $a = \frac{7}{8} + \varepsilon, b = 1 - \varepsilon$, so we know now that $\left(\frac{7}{8} + \varepsilon, 1 - \varepsilon\right)$-gap MAX E3SAT is NP-hard. Calculating the inapproximability:

$$\frac{\frac{7}{8} + \varepsilon}{1 - \varepsilon} = \frac{7}{8} - \varepsilon,$$

so we know that MAX E3SAT is $\left(\frac{7}{8} - \varepsilon\right)$-inapproximable. Note that, since we have 8 clauses representing each permutation of the three variables, if we just randomly assign the variables some values, the expectation is that only one clause will be false, so we have an expected $\frac{7}{8}$ of the clauses being satisfied.

6 Other problems

6.1 Label cover

The label cover scenario is: given a bipartite graph $G(A \dot\cup B, E)$ where we can decompose $A = A_1 \dot\cup A_2 \dot\cup \cdots \dot\cup A_k$ and $B = B_1 \dot\cup B_2 \dot\cup \cdots \dot\cup B_k$, with the contraints that $|A| = n = |B|, |A_i| = \frac{n}{k} = |B_j|$, we would like to choose subsets A', B' such that $A' \subseteq A, B' \subseteq B$, where $\dot\cup$ indicates a disjoint set union.

We also create a superedge (A_i, B_j) if at least one edge is in $A_i \times B_j$: an edge that connects some vertex in A_i with some vertex in B_j. This superedge is "covered" if and only if $A' \times B'$ intersects with $A_i \times B_j$.

6.1.1 Max rep

The max rep subproblem is to choose exactly one vertex from each group such that $|A' \cap A_i| = |B' \cap B_i| = 1$ for all A_i and B_i (where the i's are just indices, and do not represent a correlation between the subsets of A and B).

Our goal here is to maximize the number of edges in $A' \times B'$, which directly correlates to maximizing the number of covered superedges.

6.1.2 Min rep

This is the dual of max rep: we now allow multiple vertices from each group, but constrain that every superedge must be covered.

The goal here is to minimize $|A'| + |B'|$.

6.1.3 Special cases and related problems

The lecture notes briefly describe some special cases in label cover, as well as various levels of hardnesses. Most noteworthy are the Directed Steiner forest and Node-weighted Steiner tree problems, which were simply mentioned in lecture (and are diagrammed in the lecture notes).

7 Unique games

This is a special case of max rep. The premise is that the edges enforce a match between subsets A_i and B_j: choosing a value from one set, e.g. A_i, forces a choice in the other, e.g. B_j.

Unique games conjecture: The $(\varepsilon, 1-\varepsilon)$-gap Unique games problem is NP-hard.

The accuracy of this statement is currently under much debate (a.k.a. no one knows). However, we'd like it to be true, since it makes life easier.

The use of semidefinite programming (SDP) has proven to be the best approximation technique for these problems.