Topics in Theoretical Computer Science April 1, 2014

Lecture 7

Lecturer: Ola Svensson *Scribes: Kasun Samarasinghe and Guilhem Pujol*

1 Introduction

In the two last sessions, we saw the PCP-theorem, which states that **NP**= **PCP**$(\log(n), 1)$. The main point of this lecture is to use this theorem to prove hardness of approximation results.

Indeed, thousands of problems have been proven to be NP-complete, which means that finding exact solutions for them is very likely to be intractable. Therefore, when confronted which such problems, one can use two different approaches. The first one would be to establish heuristics. That being the case without any proven guarantees in general, and which behaves well on instances of the problem that are encountered in practice. The second one is to use proven approximation algorithms.

We will give a definition of an approximation algorithm . We show as well that in some cases, the PCP-theorem allows us to prove that there exists no such algorithms that perform well on a given NP-hard problem, unless **P**=**NP**.

2 Approximation algorithms

Definition 1 *An α-approximation algorithm for a given optimization problem is an algorithm which can be run in polynomial time and which outputs a solution S such that:*

- $\frac{cost(S)}{cost(Optimal\ solution)} \leq \alpha$ *if the problem is a minimization problem ;*
- $\frac{profit(S)}{profit(Optimal\ solution)} \geq \alpha$ *if the problem is a maximization problem.*

It is clear that we will have $\alpha > 1$ for minimization problems and $\alpha < 1$ for maximization problems.

It is possible to prove hardness of approximation results without using the PCP-theorem, as we are going to see in the two following examples.

2.1 Hardness of travelling salesman problem (TSP)

An instance of TSP is a graph where each edge is labeled with a cost, that must be a positive or a null real number. The expected output is a Hamiltonian path of minimal total cost in this graph.

What we are going to prove is that TSP falls into the worse category of approximation, that is:

Theorem 2 *For any $\alpha \geq 1$, it is NP-hard to get an α-approximation of TSP.*

Proof To prove this, we are going to introduce HAM, which is the problem of determining whether a graph has an hamiltonian cycle. We will use the fact that HAM is NP-hard (we can notice that this directly implies that TSP is NP-hard, since TSP asks for an Hamiltonian cycle in the graph). What we need is a **gap-introducing reduction** from HAM to TSP, that is, a reduction from HAM to TSP. Namely, from a graph G given as input to HAM, it builds a graph G' such that:

- if G has an Hamiltonian cycle, then G' has an Hamiltonian cycle with "low" cost.
- if G has non Hamiltonian cycle, then all Hamiltonian cycles of G' have "high" cost.

The main idea is that knowing a sufficiently good approximation of the solution of TSP for an input that we know to be in either the "high cost" or "low cost" category will enable us to determine in which of the two categories it belongs. This means that a good approximation for TSP can be used to solve HAM because it can distinguish between the two kind of outputs of our reduction.

To do this, we choose a constant M (that we will want to be very high) and we define our reduction as follows:

- The set of nodes of G' is equal to the set of nodes of G.
- G' is complete (all nodes are connected by an edge)
- The weight of an edge e of G' is 1 if e is also an edge of G, and is M otherwise

We denote by n the number of nodes in G and G'. If there exists a Hamiltonian cycle in G, then the same cycle has cost n in G'. Otherwise, any Hamiltonian cycles in G' use at least one edge that does not exist in G, thus it has a cost greater than M.

Now, suppose that we have an $\frac{M}{n}$-approximation of TSP. We denote by S the solution given by our approximation with input G'. If G has a Hamiltonian cycle, then the optimal solution has cost n. As such the cost of S is less than M. Otherwise, the optimal solution has cost greater than M, which means that S also has a cost greater than M. Therefore, we can decide whether G had a Hamiltonian cycle by looking at the cost of S, so our algorithm solves HAM. If M can be arbitrary, we can choose it as large as we want, which proves our theorem. ■

Observation 3 *Although we did not need the PCP-theorem to conclude, we were somehow "lucky" to be able to manipulate the costs of the edges at will. This trick is not a general method since many NP-hard problems, such as looking for the largest clique in a graph are purely combinatorial.*

Observation 4 *We can sometimes get positive results about existence of approximations. For instance, consider the modified version of TSP where the costs along the edges are required to satisfy the triangular inequality, i.e., for $u, v, w \in V$, $cost(u, w) \leq cost(u, v) + cost(v, w)$. The convention is that the cost between two nodes is infinite if there are no edges between them. This problem remains NP-hard. However, a well-known linear 2-approximation consists in performing a depth-first traversal of a minimum spanning tree of the graph (a sub-graph of G, which is a tree and which contains all nodes of G, and for which the sum of the costs of all edges is minimal among all spanning tree of G). A polynomial 1.5-approximation have also been found, and the theoretical limit has not been reached yet.*

2.2 Maximize accept probability

PCP verifiers, themselves, are a source of NP-hard problems. Given a $\mathbf{PCP}(\log(n), 1)$ verifier V for SAT, and an input φ, one can wonder which proof has a maximal probability of being accepted by V along with input φ.

Theorem 5 *It is NP-hard to approximate this problem within a factor of $\frac{1}{2}$.*

Proof The proof of this assertion is very simple, because the gap we need has already been introduced by the definition of a PCP verifier. Let's assume that we have a $\frac{1}{2}$-approximation for this problem. Let Π be the proof constructed by our algorithm with input φ, and p be the probability that Π is accepted by V along with input φ. We can evaluate p in polynomial time by feeding V with all possible random strings since it asks only for $O(\log(n))$ random bits. If $p \geq \frac{1}{2}$, then by the soundness property of V, φ is satisfiable. Otherwise, we have $p < \frac{1}{2}$ which means that the optimal proof cannot have a probability greater than $2p$ of being accepted, and $2p < 1$. By the completeness property of V, φ cannot be satisfiable.

Therefore, running our approximation algorithm on φ, along with an additional polynomial computation tells us whether φ is satisfiable, which means that our verifier solves SAT. ■

3 Hardness of Clique

Here we study the hardness of approximation of the maximum clique problem (or simply the clique problem). The clique problem consists in finding a maximum cardinality clique (i.e., a subset of vertices which forms a complete graph) given an undirected graph $G(V, E)$. We show here that there exists a constant $\epsilon > 0$, such that it is NP-hard to approximate the clique problem within a factor of $\frac{1}{N^\epsilon}$. We start by studying a gap introducing reduction from the SAT problem to a graph which is used to argue on the hardness results.

Lemma 6 *For fixed constants b and q, there is a gap introducing reduction that transforms, in polynomial time, a SAT formula φ of size n to a graph $G(V, E)$, where $N = |V| = n^b 2^q$ such that;*

- ***Completeness**: If φ is satisfiable, $OPT(G) \geq n^b$.*
- ***Soundness**: If φ is not satisfiable, $OPT(G) < \frac{n^b}{2}$.*

Proof Let F be a $\mathbf{PCP}(log\ n, 1)$ verifier for SAT that requires $b\ log\ n$ random bits and reads q bits from the proof.
We transform SAT instance φ into G[1] as follows.

- **Vertices**: For each couple of random string r of $b\ log\ n$ bits and truth assignment τ for q boolean variable, there is a vertex $V_{r,\tau}$ in G.
- **Edges**: Let $Q(r)$ represent the q positions in the proof that F queries given the "random string" r. Now, two vertices V_{r_1,τ_1} and V_{r_2,τ_2} are *consistent*, if τ_1, τ_2 agree on the values of the proof where $Q(r_1)$ and $Q(r_2)$ overlap. Further, we say that $V_{r,\tau}$ is *accepting* if F accepts, given a "random string" r and read τ in the proof. If two vertices are both *consistent* and *accepting*, there is an edge between them.

For example, consider the FGLSS graph defined for 2 random bits and 2 query strings illustrated in Figure 1. Vertices in the graph are labeled as $V(randombits)(querybits)$. Edges thus correspond to the vertices which are both consistent and accepting.
We will argue on the completeness and soundness based on the FGLSS graph.

- **Completeness**: According to the definition, we know that if φ is satisfiable there exists a proof Π such that F accepts with probability 1. Let $S = \{V_{r,\tau} \mid \tau\ consistent\ with\ \Pi\}$. By the definition of the transformation, for every "random string" r, there exists one vertex $V_{r,\tau} \in S$. Therefore, the cardinality of the set S becomes the number of random bits. n^b. We can see that S forms a clique since for each $V_{r_i,\tau_i}, V_{r_j,\tau_j} \in S$ they are accepting vertices as well as consistent (due to the definition of S).
- **Soundness**: Again by the definition of the verifier, if φ is not satisfiable, for all the proofs, F accepts with probability $< \frac{1}{2}$.

  Let C be a clique in G. Since all pairs of vertices in C are consistent, we know that C contains at most one vertex associated with each random string. This also implies that it is possible to build a proof Π that is consistent with every vertex of C: each vertex of C forces the value of up to k bits in the proof but the fact that the vertices are consistent ensures that there will be no contradiction. Some bits of Π might not be specified by any vertex, i.e., we can freely decide regardless of their value. For instance, we can set them to 0.

[1] This graph is known as FGLSS graph

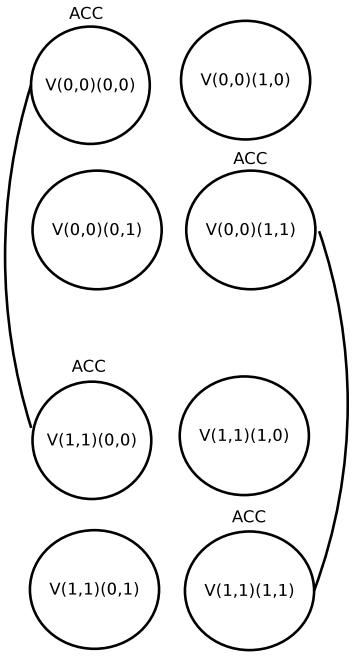

Figure 1: FGLSS graph for $b = 2$ and $q = 2$ (only a subset of the vertices and edges are drawn).

Now, we can observe that Π is accepted with probability at least $\frac{|C|}{n^b}$, because each vertex in C corresponds to a different random string that will make our verifier accept C. Because of the soundness of the verifier, this proves that $|C| < \frac{n^b}{2}$.

■

Let's consider the family $\mathbf{PCP}_{1,s}(r, q)$ of problems that have a $\mathbf{PCP}(r, q)$ verifier with *completeness* 1 and *soundness* s. We know that $\mathbf{NP}$=$\mathbf{PCP}_{1,s}(log\ n, 1)$ for any constant s. Now if we assume $\mathbf{NP}$=$\mathbf{PCP}_{1,\frac{1}{n}}(log\ n, log\ n)$ then we would have $\frac{1}{N^\epsilon}$ hardness for the clique problem.
Let V be a $\mathbf{PCP}_{1,\frac{1}{n}}(log\ n, log\ n)$ verifier for SAT that reads $b'\ log\ n$ random bits and queries $g'\ log\ n$ bits from the proof. This leads to a FGLSS graph of size $|V| = 2^{b'\ log\ n}.2^{q'\ log\ n} = n^{b'+q'}$
Now we can define the completeness and soundness of the verifier as follows.

- **Completeness**: If φ is satisfiable, G has a clique of size $n^{b'}$ (since completeness is 1).
- **Soundness**: If φ is unsatisfiable, G has no clique of size $\dfrac{n^{b'}}{n}$ (since soundness is $\dfrac{1}{n}$).

Accordingly, we can claim that clique is NP-hard to approximate within a factor of $\dfrac{1}{n}$, where $n = N^{b'+q'}$. Therefore, if we set $\epsilon = \frac{1}{q=b}$, we have the result that we exposed at the beginning of this section. Note that the soundness is due to the same argument as in the proof of Lemma 6, that n^b consistent vertices with the proof forms a clique in FGLSS graph.

Now we are left with proving $\mathbf{NP}$=$\mathbf{PCP}_{1,\frac{1}{n}}(log\ n, log\ n)$. We construct a $\mathbf{PCP}_{1,\frac{1}{n}}(log\ n, log\ n)$ verifier V for SAT. Note that our aim here is to achieve the soundness without requiring more random bits. By PCP theorem we know that there exists a $\mathbf{PCP}_{1,\frac{1}{2}}(log\ n, 1)$ verifier F for SAT that reads $b\ log\ n$ random bits and querying q bits from the proof. Here we use a result from random walks on expander

graphs (which is studied in Homework 2 Problem 1). Consider a constant degree expander graph H with n^b vertices corresponding to each random string, which are labeled by a bit string of $b\ log\ n$ bits.

Verifier V will perform a $k\ log\ n$ step random walk on H, where k is constant. Then we need $O(log\ n)$ random bits since graph is of constant degree. It will execute F on all random strings, that the vertices in the path are labeled with. V accepts if and only if F has accepted on every run. The completeness and soundness of verifier V can be concluded as:

- **Completeness**: If φ is satisfiable, there exists a proof τ_1 so that F accepts with probability 1. Consequently, it will also be accepted by V with probability 1. Hence the completeness is 1.
- **Soundness**: If φ is unsatisfiable; lets consider the set S of proofs F is going to accept. Due to the soundness we know that $|S| < \frac{n^b}{2}$. Now we know that there exists a constant k such that probability of a lazy random walk stays within S is at most $\frac{1}{n}$, since it is on H which is a constant degree expander graph. Therefore we achieve soundness without increasing the randomness.

4 Exercises

We are going to solve two exercises. One aims at improving the proof that Clique is hard to approximate within a factor $\frac{1}{2}$, and the other will give us an hardness of approximation result on MAX-k-FUNCTION SAT.

Exercise 1: Show that it is NP-hard to approximate the Clique problem within any constant factor.

To prove that Clique cannot be approximated with a factor of $\frac{1}{2}$, we build a graph that somehow modeled the behaviour of our verifier V for SAT, and use the gap of $\frac{1}{2}$ introduced by the soundness of V to conclude. We can observe that applying V several times with independent random strings as input, amounts to using a verifier that has a better soundness. This, for us, amounts to a bigger gap, and therefore a more precise result of hardness of approximation.

More formally, we define V_i as the verifier that queries i sequences of $b\log(n)$ random bytes, and applies i times V with the given input, providing the k^{th} instance of V with the k^{th} sequence of random bytes. V_i returns "yes" if all V returned "yes", and "no" otherwise.

The completeness property is a direct consequence of the completeness of V, because for a correct input φ, V accepts regardless of that the random string is.

The soundness is improved: for a bad input φ, the probability that V' answers "yes" is smaller that $\frac{1}{2^i}$. This is a result that each instance of V has an independent probability, greater than $\frac{1}{2}$, to detect that φ is bad.

Now, we can build our graph for V' exactly as we did for V. If we replace every occurrence of the soundness of V by $\frac{1}{2^i}$ in the previous proof, we will show that it is NP-hard to approximate the Clique problem within factor $\frac{1}{2^i}$.

As we can take arbitrary high values for i, we would have proven that the approximation is hard within any constant factor.

Exercise 2: Consider the problem MAX k-FUNCTION SAT:

- Given n boolean variables x_1 ,...,x_n and m functions f_1 ,...,f_m, each of which is a function of k (a constant) of the boolean variables,
- Find a truth assignment to x_1 ,...,x_n that maximizes the number of functions satisfied.

Show that it is NP-hard to approximate MAX k-FUNCTION SAT within a factor $\frac{1}{2}$.

We still consider our verifier V for SAT, which needs $b\log(n)$ random bytes and k other from the proof, with an input φ. For any possible sequence r of $b\log(n)$ bytes and any sequence x of k bytes, we define:

$f_r(x)$ is the output of V when reading φ and the sequence r as random bytes and receiving the bytes x from the proof.

This defines n^b functions of k variables, and the total number of variables is less than $k2^{b\log(n)} = kn^b$. Each assignment of x_1 ,...,x_{kn^b} corresponds to a proof.

Let's assume that we have a $\frac{1}{2}$-approximation for MAX k-FUNCTION SAT. If φ is satisfiable, all functions can be satisfied simultaneously (because for at least one proof, V always outputs 1). As such our verifier will generate an assignment that satisfies at least half of the functions. Otherwise, any assignment of x_1 ,...,x_n cannot satisfy more than half of the functions (because V must return 0 for any proof with probability at least $\frac{1}{2}$), meaning that our verifier will not be able to generate an assignment that satisfies half of the functions or more. We can thus tell whether φ is satisfiable by comparing the output of our verifier to $\frac{1}{2}$, which amounts to solving SAT.

As a bonus, we will show that this result can be used to prove that MAX-3SAT is hard to approximate. Indeed, there exists a reduction of MAX k-FUNCTION SAT to MAX-3SAT such that if there are no assignments that leave less than half of the functions unsatisfied in MAX k-FUNCTION SAT, then every assignment leaves at least a fraction $\frac{1}{2\times k\times 2^k}$ of the clauses unsatisfied in the corresponding instance of MAX-3SAT.

The reductions works as follow:

- Each function is converted to a conjunction of at most 2^k disjunctions of literals. One way to do this is to observe that a disjunction of k literals formed with the k variables that are arguments of the function is always true, except for one particular assignment of the k variables. Therefore, for each assignment of the variable of f_i that makes f_i false, we can create one disjunction that negates it exactly. The conjunction of all these disjunctions will be a formula that is true if and only if f_i is true. It will also be composed of at most 2^k disjunctions.

- Each of these disjunctions, of at most k elements, has to be converted into a conjunctions of disjunctions of 3 elements. For each initial dis-junction $l_1 \vee$ ... $\vee l_k$, we introduce $k-2$ auxiliary variables y_1, ..., y_{k-2} and we build the equivalent formula : $(l_1 \vee l_2 \vee y_1) \wedge (\overline{y_1} \vee l_3 \vee y_2) \wedge ... \wedge (\overline{y_{k-2}} \vee l_{k-1} \vee l_k)$

We end up with less than k 2^k disjunctions of 3 literals, whose conjunction is equivalent to the assertion that all f_i are satisfied. Furthermore, for each f_i that is not satisfied, at least one of these disjunctions will be false, which gives us the desired proportion.

From this, we can conclude that it is NP-hard to approximate MAX-3SAT within a factor of $1-\frac{1}{2k2^k}$.