CMSC858F: Algorithmic Lower Bounds: Fun with Hardness Proofs
Fall 2014

Instructor: Mohammad T. Hajiaghayi
Scribe: Melika Abolhassani

October 08, 2014

1 Overview

In this lecture, "Approximation Preserving Reductions" and "Gap Preserving Reduction" are introduced, followed by some examples from APR reduction.

2 Definitions

The first definition here is **Approximation Preserving Reduction(APR reduction)**. Approximation Preserving Reduction from problem A to B in general means if we can well-approximate B, we can well-approximate A as well. Indeed, there are more than eight notions of approximation preserving reductions differing only in some details.

Definition 1 *Let* π_1 *and* π_2 *be two minimization problems. An* Approximation (factor) Preserving Reduction *from* π_1 *to* π_2 *consists of two poly-time computable functions* f, g *such that*

1. *For any instance* I_1 *of* π_1*,* $I_2 = f(I_1)$ *is an instance of* π_2 *such that* $\mathsf{OPT}_{\pi_2}(I_2) \leq \mathsf{OPT}_{\pi_1}(I_1)$.
2. *For any feasible solution* S_2 *of* I_2*,* $S_1 = g(I_1, S_2)$ *(*g *maps* S_2 *into an instance of* I_1*) we have* $\mathsf{Cost}_{\pi_1}(I_1, S_1) \leq \mathsf{Cost}_{\pi_2}(I_2, S_2)$.

Note that $\mathsf{OPT}_{\pi_2}(I_2) \leq \mathsf{OPT}_{\pi_1}(I_1) \leq \mathsf{Cost}_{\pi_1}(I_1, S_1) \leq \mathsf{Cost}_{\pi_2}(I_2, S_2)$. Therefore, if there is an approximation factor Δ for π_2 then there is an approximation factor Δ for π_1 as well.

$$\frac{\mathsf{Cost}_{\pi_2}(I_2, S_2)}{\mathsf{OPT}_{\pi_2}(I_2)} \leq \Delta \Rightarrow \frac{\mathsf{Cost}_{\pi_1}(I_1, S_1)}{\mathsf{OPT}_{\pi_1}(I_1)} \leq \Delta \tag{1}$$

Similarly, if there is no Δ-approximation for π_1, then there is no Δ-approximation for π_2.

Note : If π_1 and π_2 are maximization problems we should have $\mathsf{OPT}_{\pi_2}(I_2) \geq \mathsf{OPT}_{\pi_1}(I_1)$ and $\mathsf{Cost}_{\pi_1}(I_1, S_1) \geq \mathsf{Cost}_{\pi_2}(I_2, S_2)$.

Another useful reduction is **Gap Preserving Reduction** specially because of PCP theorem.

Definition 2 *Let P and P' be maximization problems. A* **Gap Preserving Reduction** *from P to P' is a polynomial-time algorithm which given an instance I of P with $|I| = n$ produces an instance I' of P' with size n' such that if*

1. $\mathsf{OPT}(I) \geq h(n)$ *then* $\mathsf{OPT}(I') \geq h'(n')$
2. $\mathsf{OPT}(I) \leq \frac{h(n)}{g(n)}$ *then* $\mathsf{OPT}(I') \leq \frac{h'(n')}{g'(n')}$

for some functions g, h, g', h' with $g(n) \geq 1$ and $g'(n') \geq 1$.

For minimization problems the above two conditions change to the following two conditions:

1. $\mathsf{OPT}(I) \leq h(n)$ *then* $\mathsf{OPT}(I') \leq h'(n')$
2. $\mathsf{OPT}(I) \geq h(n)g(n)$ *then* $\mathsf{OPT}(I') \geq h'(n')g'(n')$

Observe that if $\mathsf{Gap} - \mathsf{P}_{g(n)}$ is hard and thus approximating P within factor $g(n)$ is hard, then $\mathsf{Gap} - \mathsf{P}'_{g'(n')}$ is also hard (and thus approximating P within factor $g'(n')$ is hard)

3 PCP, Unique Games Conjecture and Other Theorems

Theorem 1 *PCP Theorem{Raz'98}: For every $\epsilon > 0$, there is an instance of Max-Rep with $n \leq \frac{1}{\epsilon^{O(1)}}$ (recall that in Max-Rep we have $2n$ sets of k cardinality each) such that it is NP-hard to distinguish between the following two cases:*

1. *There is a solution which covers all super-edges.*
2. *In every solution we can cover at most ϵ fraction of super-edges.*

Theorem 2 *Unique Games Conjecture{Khot'02}: For every $0 < \epsilon < \frac{1}{2}$, there is a Unique Games instance such that it is NP-hard to distinguish between the following two cases:*

1. *There is a solution which covers at least $1 - \epsilon$ fraction of super-edges.*
2. *In every solution we can cover at most ϵ fraction of super-edges.*

Note that if *UGC* is correct (though there is some doubt about it at least in this very strong form), then every gap-preserving reduction/approximation preserving reduction using that gives us inapproximability results. Proving inapproximability is often done as follows:

$$\text{UGC, PCP} \xrightarrow[\text{reduction}]{\text{Gap-preserving}} \text{some problems} \xrightarrow{\text{APRreduction}} \text{other problems}$$

A fundamental and quite difficult result which is equivalent to PCP theorem (and in fact derived from it) is as follows.

Theorem 3 *: There exists constant ϵ_0 such that it is NP-hard to distinguish the following two cases for* Max-3SAT *problem:*

1. *The given input instance is satisfiable.*
2. *At most $1 - \epsilon_0$ fraction of the clauses are satisfiable in every assignment.*

Theorem above says Max-3SAT is *APX-hard*. Using this theorem we can get $\frac{7}{8}$-inapproximability result for 3-Sat but then the gap is not between all satisfiability and $\frac{7}{8}$-satisfiability.

Another useful consequence of PCP is the following theorem :

Theorem 4 *: Unless $\mathsf{NP} \subseteq \mathsf{DTIME}(O(n^{polylog(n)}))$, it is hard*

1. *For Max-Rep to distinguish the following two cases*
   (a) *We cover all super edges*
   (b) *We can cover at most $\frac{1}{2^{\log^{1-\epsilon}(n)}}$ fraction of super-edges.*
2. *For Min-Rep to distinguish the following two cases*
   (a) *There is a solution of size $2k$(i.e. exactly one vertex from each set).*
   (b) *To cover all super-edges we need at least $2k2^{\log^{1-\epsilon}(n)}$ fraction of super-edges.*

Note that by the above theorem both Min-Rep and Max-Rep cannot be approximated within ration $2^{\log^{1-\epsilon}(n)}$ for any fixed $\epsilon > 0$ unless $\mathsf{NP} \subseteq \mathsf{DTIME}(O(n^{polylog(n)}))$.

Theorem 5 *If it is NP-hard for Max-3SAT problem to distinguish between these two cases*

1. *Satisfy all clauses.*
2. *Satisfy only a c-fraction of the clauses.*

Then there is no approximation factor better than $\frac{1}{c}$ for Max-3SAT.

Proof: Say there is an α-approximation algorithm such that $\alpha > \frac{1}{c}$. Run this algorithm on Max-3SAT instance. If we are in case 1 that is all the clauses can be satisfied, the algorithm would satisfy more than $c - fraction$. Therefore, if less than c-fraction of the clauses are satisfied we are in case 2. Thus, we are in case 1 iff more than c-fraction of the clauses are satisfied.

■

Scribe: Melika Abolhassani

Date: 10/08/2014

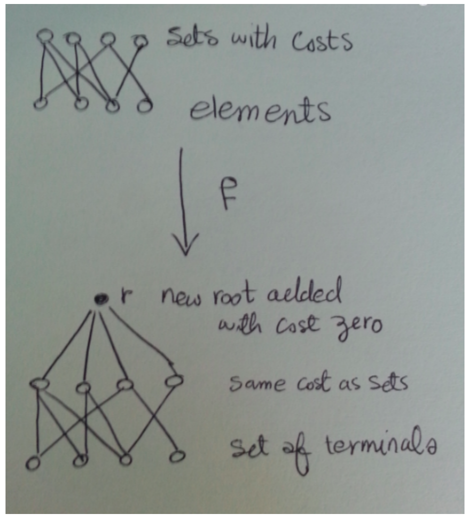

Figure 1: APR reduction from Set Cover to Node Weighted Steiner Tree

4 APR reduction Examples

4.1 Example 1: APR reduction from Set Cover to Node Weighted Steiner Tree

Definition 3 Node Weighted Steiner Tree *Given graph* $G = (V, E)$ *with weights on its nodes, a subset of* V *marked as terminal and a node* $r \in V$ *as root, We want to choose a set of nodes of minimum weight such that all the terminals are connected to the root.*

Reduction: Construct a graph from the given set cover instance I_1 as follow. Consider a vertex corresponding to each set with same weight. Add a vertex corresponding to each element with weight zero. Connect each element to the sets it belongs to. Add a root with weight zero connected to all sets. See figure 1 for illustration. The vertices corresponding to the elements are the set of terminals. This is an instance I_2 of Node Weighted Steiner Tree. Note that $I_2 = f(I_1)$ if we assume f to be the algorithm for constructing this graph. Assume we have the optimum solution to I_1. Then choose the corresponding nodes to the sets in I_2 along with the set of terminals and the root. Each element is at least in one chosen set and therefore, each terminal is at least connected to one of root's neighbors. This gives a solution of the exact same cost in I_2 as optimum of I_1; i.e, $\mathsf{OPT}_{\pi_2}(I_2) \leq \mathsf{OPT}_{\pi_1}(I_1)$.

Now suppose you have a solution S_2 of I_2. Similarly, choose all the sets corresponding to the chosen vertices to obtain solution $S_1 = g(I_1, S_2)$ of I_1. With the same reasoning we can argue that S_1 is a valid solution for set cover instance and $\mathsf{Cost}_{\pi_1}(I_1, S_1) = \mathsf{Cost}_{\pi_2}(I_2, S_2)$. This completes the reduction.

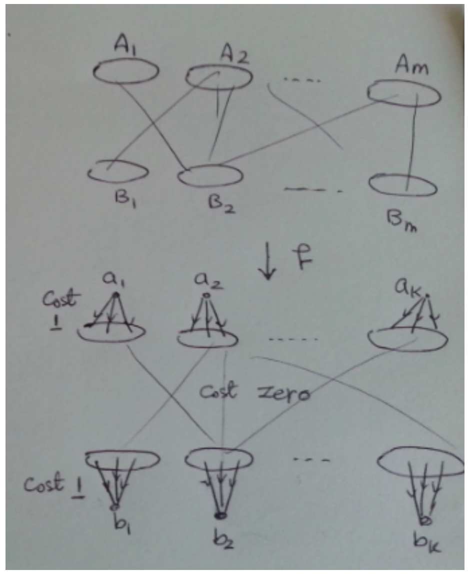

Figure 2: APR reduction from Min Rep to Directed Steiner Forest

Since there is no $\log n$-approximation for Set Cover problem, there is no $\log n$-approximation for Node Weighted Steiner Tree as well.

Note : There is no APR reduction from Min Rep to *Edge Weighted Steiner Forest* from. In fact Edge Weighted Steiner Forest problem has 1.39-approximation.

4.2 Example 2: APR reduction from Min Rep to Directed Steiner Forest

Assume you have the Min Rep instance graph. Direct all the edges from set $A = \cup_{i=1}^{k} A_i$ to set $B = \cup_{i=1}^{k} B_i$ with weight 0 to get a weighted directed graph. Also add $2k$ nodes (where k is the number of sets in each part of the Min Rep instance) $a_1, a_2, \ldots, a_k$ and $b_1, b_2, \ldots, b_k$ such that for any $i \in [k]$, a_i has a directed edge to all vertices in A_i with weight 1 and all vertices in B_i have directed edges to b_i with weight 1. Figure 2 illustrates this reduction. Require all pairs (a_i, b_j) in directed steiner forest instance iff there is a superedge between A_i and B_j. It is easy to see that for each solution to an instance of Min Rep, we have an exact same cost solution to the corresponding Directed Steiner Forest instance and vice versa.

4.3 Example 3: APR reduction from Dominating Set to Set Cover

Let I_1 be an instance of the Dominating Set problem. Build I_2, an instance of Set Cover problem, by considering one element and one set for each vertex

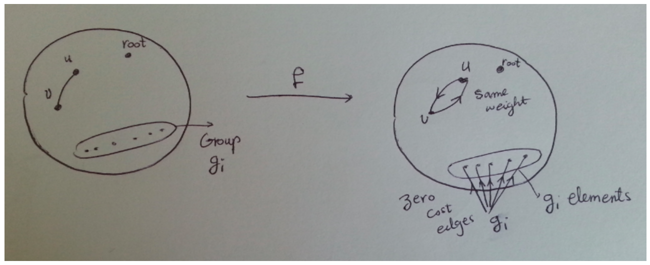

Figure 3: APR reduction from Group Steiner Tree to Directed Steiner Tree

in I_1. The set corresponding to vertex v in I_1 instance contains all the elements corresponding to v and its neighbors and has the same cost as the vertex weight. For an optimal solution of Dominating Set problem, choose all the sets corresponding to the chosen vertices to obtain a solution of I_2 of the exact same cost. Similarly, a solution to I_2 can be changed to a same cost solution of I_1 by choosing the vertices corresponding to the chosen sets in I_2.

Dominating Set problem is $\log n$-hard to approximate and so is Set Cover problem by this reduction.

4.4 Example 4: APR reduction from Group Steiner Tree to Directed Steiner Tree

An instance I_2 of Directed Steiner Tree is constructed from an instance I_1 of Group Steiner Tree as follows. Direct all the edges of the graph G of I_1 in both directions with the same weight as the undirected graph. The root remains the same. Add vertex g_i for each group i in G and connect it with a directed zero-cost edge to all vertices in group i and add g_i to the set of terminals. Figure 3 shows how the reduction is done. The optimal solution in I_1 can be changed into a same cost solution of I_2 by choosing the same set of edges plus all the edges originating from g_i for any group i. On the other hand, any solution of I_2 has at least one vertex of group i connected to the root and thus is a same cost solution of I_1 as well. This completes APR reduction.

4.5 Example 5: APR reduction from Max-3SAT to Independent Set

Let ϕ be an instance of the Max-3SAT problem. Let $C_1, \ldots, C_m$ be the set of clauses in ϕ and $x_1, x_2, \ldots, x_n$ be the set of variables in ϕ. Construct graph G_ϕ as follows. Add three nodes per clause (one node for each literal in the clause) and connect them to get a triangle for each clause. We also add an edge between two nodes corresponding to x_i and $\bar{x_i}$ if they are from two different clauses. In this setting, $\mathsf{OPT}_{Max-3SAT}(\phi) = \mathsf{OPT}_{Ind\ Set}(G_\phi)$. Figure 4 shows edges for two clauses.

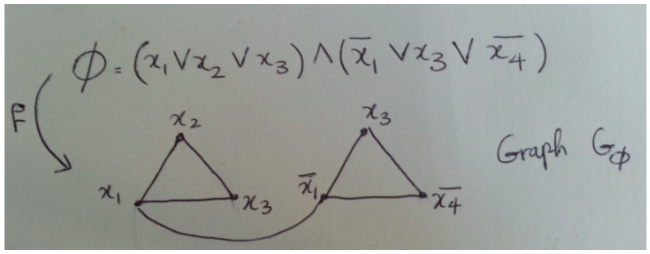

Figure 4: APR reduction from Max-3SAT to Independent Set

Since according to PCP, Max-3-SAT is APX-hard, so is independent set. Also the reduction implies that there is a gap between $\mathsf{OPT} = m$ and $\mathsf{OPT} < (1 - \epsilon_0)m$ for independent set and thus it is a gap-preserving reduction as well.