Theory of Computer Science to Msc Students, Spring 2007

Lecture 4

Lecturer: Dorit Aharonov *Scribe: Ram Boukobza & Yair Yarom*

Revised: Shahar Dobzinski, March 2007

1 FPTAS - Fully Polynomial Time Approximation Scheme

In the last lecture we have seen a $(1-\epsilon)$-approximation algorithm for the Knapsack problem. We have mentioned that this algorithm is an FPTAS for this problem. Let us now formally define what an FPTAS is:

Definition 1 (FPTAS) *An algorithm $\mathcal{A}$ is an FPTAS for an optimization problem P, if given an input I for P and $\epsilon > 0$, $\mathcal{A}$ finds in time polynomial in the size of I and in $\frac{1}{\epsilon}$, a solution S for I that satisfies*

$$|val(I) - val(S)| \leq \epsilon val(I),$$

where $val(I)$ is the optimal value of a solution for I.

In the last lecture we saw a FPTAS for the knapsack problem, is there a FPTAS for all NP-Complete problems? As we shall see next, the answer is no (unless $P = NP$).

Definition 2 (SNP-C: Strong NP-Complete) *Given a search problem Pi over the positive integers and a polynomial p, denote by Pi_p the restriction of Pi to instances I such that $val(I) \leq p(length(I))$ ($length(I)$ is the length of the input instance I). We say that Pi is NP-hard in the strong sense if there is a polynomial p over the integers such that Pi_p is NP-hard. Pi is strongly NP-complete if it is NP-hard in the strong sense and the corresponding decision problem is in NP.*

Observation 3 *A problem in SNP-C does not have a FPTAS unless $P = NP$.*

Proof Assume that a problem $Pi \in SNP - C$ has a $FPTAS$, $\mathcal{A}_{Pi}$. By our assumption, there is a polynomial p over the integers such that Pi_p is NP-hard. Given an instance I for Pi_p take $\epsilon = \frac{1}{2p(length(I))}$ and run $\mathcal{A}_{Pi}$ on the pair I, ϵ. Since $\mathcal{A}_{Pi}$ is a FPTAS for Pi, it returns a solution, S, that satisfies

$$|val(I) - val(S)| \leq \epsilon val(I) = \frac{val(I)}{2p(length(I))} \leq 1/2.$$

Since $val(I)$ and $val(S)$ are both integers we must have $val(I) = val(S)$. On the other hand, $\mathcal{A}_{Pi}$ is polynomial in $length(I)$ and $1/\epsilon$ which in this case is bounded by a polynomial in $length(I)$. So, we have solved Pi_p, an NP-hard problem, in polynomial time, concluding $P = NP$. ■

We know that there are problems in SNP-C (e.g., Vertex Cover, MAX-SAT, and most other NP-complete languages). As a direct result, we get that when considering approximations, not all NP-Complete languages are equal.

2 Rounding Applied to Set Cover

Let $E = \{e_1, \dots, e_n\}$ be a set of elements, and let $S = \{S_1, \dots, S_k\} \subseteq 2^E$. A set C is called a *cover* for E if for every $e \in E$ there is a set $S \in C$, such that $e \in S$. The *Set Cover* problem asks to find a cover $C \subseteq S$ for E with minimal size.

Define the frequency of an element e_i to be the number of sets $S_i \in S$ it is in. Denote by f the frequency of the most frequent element.

Observation 4 *The Vertex Cover problem is a special case of the Set Cover problem , where the elements are the edges of the graph, and S_i is the set of edges that vertex i is incident in. In this case $f = 2$ since each edge (u, v) belongs to exactly 2 S_i's: S_u and S_v.*

We will now present an f-approximation to the Set Cover problem. We first present Set Cover as an program. Then,we relax the integer constraints and solve the relaxed problem using linear programming. Finally, we convert the LP solution to an integer solution using rounding and obtain and prove that it provides an f-approximation to the original problem.

For each $S_i \in S$ define a variable X_{S_i}. The integer program is as follows:

Minimize: $\Sigma_{S_i \in S} X_{S_i}$

Subject to:

- For each element $e \in E$: $\Sigma_{S_i | e \in S_i} X_{S_i} \geq 1$
- For each S_i: $X_{S_i} \in \{0, 1\}$

The first type of constraints requires that every item is covered, and the second type of constraints are the integer constraints. Clearly, every solution to the Set Cover problem is also a solution to the integer program (with the same value), and vice versa. Denote by OPT the optimal solution. Unfortunately, since the SET-Cover problem is NP-Complete, this makes integer programming NP-complete. To (partially) overcome this, we relax the integer constraints and obtain the Linear Programming relaxation of the problem:

Minimize: $\Sigma_{S_i \in S} X_{S_i}$

Subject to:

- For each element $e \in E$: $\Sigma_{S_i | e \in S_i} X_{S_i} \geq 1$
- For each S_i: $0 \leq X_{S_i} \leq 1$

Notice that we can solve the linear relaxation in polynomial time. Let OPT* be the optimal solution of the LP Problem.

The next step of the algorithm takes the solution of the linear program (LP), and rounds it back to an integer solution. In the current algorithm, the rounding is quite simple: take each variable X_{S_i}, if $X_{S_i} \geq \frac{1}{f}$ choose S_i to be in the set cover (i.e., set $X_{S_i} = 1$ in the integer program). Otherwise, S_i is not in the set cover ($X_{S_i} = 0$).

Let $COVER = \{S_i | X_{S_i = 1}\}$ be the solution obtained. We now show that COVER is indeed a valid set cover.

Claim 5 *Cover is a set cover.*

Proof Assume by contradiction that there exists $e \in E$, which is not covered by COVER. Thus $\forall_{S_i : e \in S_i} X_{S_i} < \frac{1}{f}$ (otherwise X_{S_i} would be in COVER and e would be in the set cover). However, this cannot be the case since by the LP constraints $\Sigma_{S_i | e \in S_i} X_{S_i} \geq 1$. Recall that e is incident in at most f sets. Hence, there must be a set S_i where the initial value in the LP of X_{S_i} was at least $\frac{1}{f}$. A contradiction. ■

The next claim shows that the algorithm indeed provides an approximation ratio of f. We use the fact that $OPT^* \leq OPT$ (which is true because we have just added constraints in the integer program comparing to the linear program).

Claim 6 $|COVER| \leq |OPT^*| \cdot f$

Proof The claim is true because the value of each X_{S_i} cannot increase by a factor of more than f in the integer solution we construct. ■

3 Randomized Rounding Applied to MAX-SAT

Recall the MAX-SAT problem where we are given a SAT formula ϕ $C_1 \wedge ... \wedge C_m$ on variables $X_1, ..., X_n$. For an assignment A we denote by $\phi(A)$ the number of clauses that are satisfied in A. As in our approximation algorithm for SET-COVER, we will round the linear relaxation of the problem. However, this time we will use *randomized* rounding.

The integer program we present now is less straightforward than the previous one. For each variable X_i in the formula, assign a variable y_i. The meaning of $y_i = 1$ is that $X_i = true$ assignment, and $y_i = 0$ otherwise. For each clause $C_i \in C$, assign a variable z_{C_i}. The meaning is that $z_{C_i} = 1$ if and only if C_i is satisfied. The integer program:

Maximize: $\Sigma_{C_i} z_{C_i}$

Subject to:

- For each C_i: $\Sigma_{i \in C_i^+} y_i + \sum_{i \in C_i^-} (1 - y_i) \geq z_{C_i}$
- For each clause C_i: $C_i \in \{0, 1\}$
- For each variable X_i: $y_i \in \{0, 1\}$

Where C_i^- is the set of variables in clause C_i that are negated, and C_i^+ is the set of variables that are not negated. These constraint forces at least one literal to be true in order for the clause C_i to be satisfied. Let OPT be the optimal solution. Similarly to before, the LP relaxation is:

Maximize: $\Sigma_{C_i} z_{C_i}$

Subject to:

- For each C_i: $\Sigma_{i \in C_i^+} y_i + \sum_{i \in C_i^-} (1 - y_i) \geq z_{C_i}$

- For each clause C_i: $0 \leq C_i \leq 1$
- For each variable X_i: $0 \leq y_i \leq 1$

Given the LP solution $OPT* = (y*, z*)$, we set we assign $X_i = true$ with probability y_i^*.

The following claim proves a lower bound on the probability that a clause C_i is satisfied using the randomized rounding solution.

Claim 7 *Let k be the maximum number of literals in some clause C_i. Let $\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k$. The probability that C_i is satisfied by the randomized rounding procedure is at least $\beta_k z_c^*$.*

Proof We prove the claim for the case that all literals in c are not negated. let $c = x_1 \vee \ldots \vee x_k$,

$$\begin{aligned}
\Pr(C_i \text{ is satisfied}) &= 1 - \Pr(C_i \text{ is not satisfied}) \\
&= 1 - \Pr(\text{all } X_i\text{'s are false}) \\
&= 1 - \Pi_{i=1}^k \Pr(X_i \text{ is false}) \\
&= 1 - \Pi_{i=1}^k (1 - y_i^*), \text{ (since } \Pr(X_i = true) = y_i^*) \\
&\geq 1 - \left(\frac{\sum_{i=1}^k (1 - y_i^*)}{k}\right)^k \\
&= 1 - \left(1 - \frac{\sum_{i=1}^k y_i^*}{k}\right)^k \\
&\geq 1 - \left(1 - \frac{z_c^*}{k}\right)^k, \text{ (from the LP constraints)}
\end{aligned}$$

where the first inequality follows from arithmetic-geometric mean inequality: $\frac{a_1 + \ldots + a_k}{k} \geq \sqrt[k]{a_1 \times \ldots \times a_k}$.

Define $g_1(z) = 1 - \left(1 - \frac{z}{k}\right)^k$. Note that $g_1(z)$ is a concave function, and that $g_1(0) = 0$ and $g_1(1) = \beta_k$. Define $g_2(z) = \beta_k z$. Note that $g_2(z)$ is a linear function also with $g_2(0) = 0$ and $g_2(1) = \beta_k$. Therefore, for $z \in [0,1]$ $g_1(z) \geq g_2(z) = \beta_k z$. And since $z^* \in [0,1]$ we have $g_1(z^*) \geq \beta_k z^*$. Hence, $\Pr(C_i \text{ is satisfied}) \geq g_1(z^*) \geq \beta_k z^*$. ■

Next, we prove a lower bound on the expected number of clauses satisfied by the randomized rounding solution. Let T_{C_i} be an indicator random variable that gets the value of 1 if and only if the clause C_i is satisfied (and 0 otherwise). By the previous claim:

$$\begin{aligned}
E\left[\text{number of satisfied clauses}\right] &= E\left[\Sigma_{C_i} T_{C_i}\right] \\
&= \Sigma_{C_i} E\left[T_{C_i}\right] \\
&\geq \Sigma_{C_i} \beta_k z_{C_i}^*, \\
&= \beta_k \Sigma_{C_i} z_{C_i}^* \\
&= \beta_k OPT^*
\end{aligned}$$

Observe that as usual we have that $OPT \leq OPT^*$. Finally, we get that $OPT \geq E\left[\text{number of satisfied clauses}\right] \geq \beta_k \cdot OPT$. We conclude that in expectation we have a $\frac{1}{\beta_k}$ approximation. Note that β_k is a decreasing function in k. Hence, as k increases, the expected approximation becomes worse and in infinity $\lim_{k\to\infty} \beta_k = 1 - \frac{1}{e}$.

Recall that the previous random algorithm for MAX-SAT, which was to set each variable to true independently with probability 1/2 gave a better expected approximation as k increased. A combined randomized algorithm, which chooses with probability 1/2 the either first or the second algorithm could give a $\frac{3}{4}$-approximation algorithm, regardless of the value of k.